Model-based Calibration of Engine Control Units Using Gaussian Process Regression

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

von
Dipl.-Ing. Nils Tietze
Geboren am 07.09.1985 in Oldenburg

Referent:	Prof. Dr.-Ing. Ulrich Konigorski
Korreferent:	Prof. Dr.-Ing. Oliver Nelles
Tag der Einreichung:	06. Februar 2015
Tag der mündlichen Prüfung:	08. Mai 2015

D17
Darmstadt 2015

Erklärung laut §9 PromO

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Datum und Unterschrift

Acknowledgements

This thesis was written during my time as a PhD student at Bosch Engineering GmbH in Abstatt and was supervised by the Technical University of Darmstadt. I would especially like to thank my group leader at Bosch Engineering GmbH, Marcel Richter for his confidence, his support, the inspiring discussions and the opportunity to conduct my studies at the Bosch Engineering GmbH. Many thanks to my advising Professor, Ulrich Konigorski, for the acceptance as a PhD candidate, the technical discussions and the formal guidance throughout my graduate work. Thanks to Professor Oliver Nelles for being the second examiner and to Professor Michael Bargende for his guidance. Thanks to my colleagues, especially Duy Nguyen-Tuong, Benjamin Hartmann, Tobias Gutjahr, Christian Fleck, Ingo Hein, and Dariusz Humienik for the collaboration. Lastly, I would like to thank my family and friends for the support over the past years. This thesis is dedicated to my father Günther Tietze, who unfortunately passed away on 26.05.2014 and whose mental support I missed during the final stages of writing this thesis.

Abstract

Reducing the number of tests on vehicles is one of the most important requirements for increasing cost efficiency in the calibration process of engine control units (ECU). Here, employing virtual vehicles for a model-based calibration of ECUs is essential. Modelling components for virtual vehicles can be a tedious and time-consuming task. In this context, data-based modelling techniques can be an attractive alternative to physical models to increase efficiency in the modelling process. Data-based models can incorporate unknown nonlinearities encoded in the sampled data, resulting in more accurate models in practice. In combination with automated measurement, data-based modelling can help to significantly accelerate the calibration process. Furthermore, the fast simulation speed of the resulting models allows their implementation into real-time simulation environments, such as Hardware-in-the-Loop (HiL) systems, and thus enables a model-based calibration of the related ECU software function. However, generating appropriate data for learning dynamic models, i.e., the transient Design of Experiments (DoE), is not straightforward, since system boundaries and permissible excitation frequencies are not known beforehand. Thus the training data of the system measurement will be inconsistent and the main challenge of the identification process is to deal with this data to achieve a globally valid model. Furthermore, when dealing with dynamic systems in an automotive context, the Engine Control Unit typically changes operating modes while driving. Thus nonlinearities and changes of physical structures appear, which need to be considered in the model. In this thesis, a modelling system called the Local Gaussian Process Regression (LGPR), is used and adapted in order to receive a flexible modelling approach, which allows an iterative modelling process and obtains robust and globally valid dynamic models. The adapted LGPR approach is employed for the ECU calibration of dynamical automotive systems, which is critical regarding system excitation. Using LGPR, it is possible to measure the system iteratively while exploring the relevant state-space regions and improving the quality of the model step by step. The results show that LGPR is beneficial for iterative modelling of dynamical systems. Compared to the traditional

Gaussian Process Regression (GPR) modelling approach, LGPR yields better results regarding the variable system dynamics.

Kurzfassung

Die Reduktion von Versuchsträgern ist eine der wichtigsten Anforderung zur Steigerung der Kosteneffizienz im Applikationsprozess (Kalibrierprozess) von Motorsteuergeräten (ECU). Durch Bereitstellung virtueller Fahrzeuge kann die Applikation der ECU modellbasiert erfolgen. Die Erstellung der einzelnen Modelle kann sich als schwierig und zeitaufwendig erweisen, wodurch sich datenbasierte Modellierungsmethoden als vielversprechende Alternative anbieten. Daten-basierte Modelle sind in der Lage, Nichtlinearitäten anhand der Systemvermessung zu berücksichtigen, wodurch erfahrungsgemäß exakte Modelle erstellt werden können. In Kombination mit einer autmatisierten Systemvermessung kann die datenbasierte Modellierung zu einer signifikanten Beschleunigung des Applikationsprozesses führen. Weiterhin ermöglicht die schnelle Simulation dieser Modelle eine Implementierung in eine Echtzeit Simulationsumgebung, wie Hardware-in-the-Loop (HiL) Systeme, und somit eine modellbasierte Applikation der zugehöhrigen Steuergerätefunktionen an diesen Systemen.
Die Herausforderung dieser Modellbildung besteht in der Erzeugung angemessener Daten, insbesondere bei dynamischen Systemen. Die Versuchsplanung zur Erzeugung angemessener Daten stellt sich als schwierig heraus, insbesondere da Systemgrenzen und zulässige Anregungsfrequenzen über das System nicht vorhanden sind. Bei der Identifikation dynamischer Systeme ist es somit die Herausforderung, inkonsistente Daten zu handhaben und dennoch ein global valides Modell zu erzeugen. Eine weitere Anforderung im automotiven Zusammenhang besteht im Auftreten von Betriebsartenumschaltungen des Motorsteuergerätes, wodurch neben Nichtlinearitäten auch Änderungen des physikalischen Systems auftreten können. Auch diese Effekte müssen berücksichtigt werden.
In dieser Arbeit wird ein Modellierungsansatz verwendet, der sich Lokale Gaußprozess Regression (LGPR) nennt. Eine Erweiterung dieses Ansatzes ermöglicht eine flexible und iterative Modellbildung, um robuste und valide dynamische Modelle zu erzeugen. Die angepasste LGPR kann für die ECU Applikation von dynamischen automotiven Systemen eingesetzt werden, welche kritisch in Bezug auf die Anregung sind. Unter

Vewendung von LGPR ist es möglich, das System iterativ zu vermessen während die relevanten Zustandsraumregionen Schritt für Schritt erfasst werden und dadruch das Modell kontinuierlich verbessert wird. Die Ergebnisse zeigen, dass LGPR vorteilhaft für die iterative Modellbildung dynamischer Systeme ist. Im Vergleich zur Standard Gaußprozess Regression (GPR) führt LGPR zu besseren Ergebnissen hinsichtlich variablen Systemdynamiken.

Contents

1 Introduction

Due to the ever increasing number of control variables in modern passenger cars and stricter market regulations, e.g. concerning emissions, the number of calibration labels in state-of-the-art Engine Control Units (ECU) increases exponentially. Counteracting this dilemma, this thesis presents a tool chain for model-based calibration and introduces an adaptation of a powerful regression algorithm for the identification of dynamic systems. The first chapter explains the motivation for this work, gives an introduction to the calibration of ECU and continues with a discussion about the state-of-the-art in model-based ECU calibration. Finally, it concludes with an overview and the contribution outline this work.

1.1 Motivation

The increase of legal requirements for exhaust emissions and system monitoring in combination with growing customer requirements regarding performance and comfort leads to a continuous increase of complexity in modern power train development of vehicles. Downsizing with a highly charged engine in combination with complex exhaust after treatment are current promising solutions to fulfill the requirements. Environmentally conscious driving is leading to a high number of vehicle variants and individual vehicle concepts. Combustion engines and electric motors are combined and allow a variation of hybrid concepts and further new functionalities. Start-Stop, recuperation and communication concepts between components are being researched. In the future, the increase in complexity will continue and the goal is to counteract this using intelligent methods to keep complex things understandable.
However, in power train development all complexity comes together into one central unit, the Engine Control Unit. The ECU controls the engine and further communicates with other control units like the Electronic Stability Control (ESP) or transmission. It ensures the functionality of the vehicle and diagnoses failure in emission relevant

components. Fig. 1.1 gives an overview of a modern, supercharged engine concept controlled by a Bosch ECU. Actuators like the *throttle device*, *camshaft position*, *waste gate position*, *spark timing* or *injection time* allow the ECU to operate the engine in optimal states. Because of missing physical information about dynamical effects, nonlinearities as well as disturbances, the ECU needs feedback about the state of the engine. Thus, many sensors like the *air mass meter*, *position sensor*, *knock sensor* or *oxygen sensor* are needed to determine engine states.

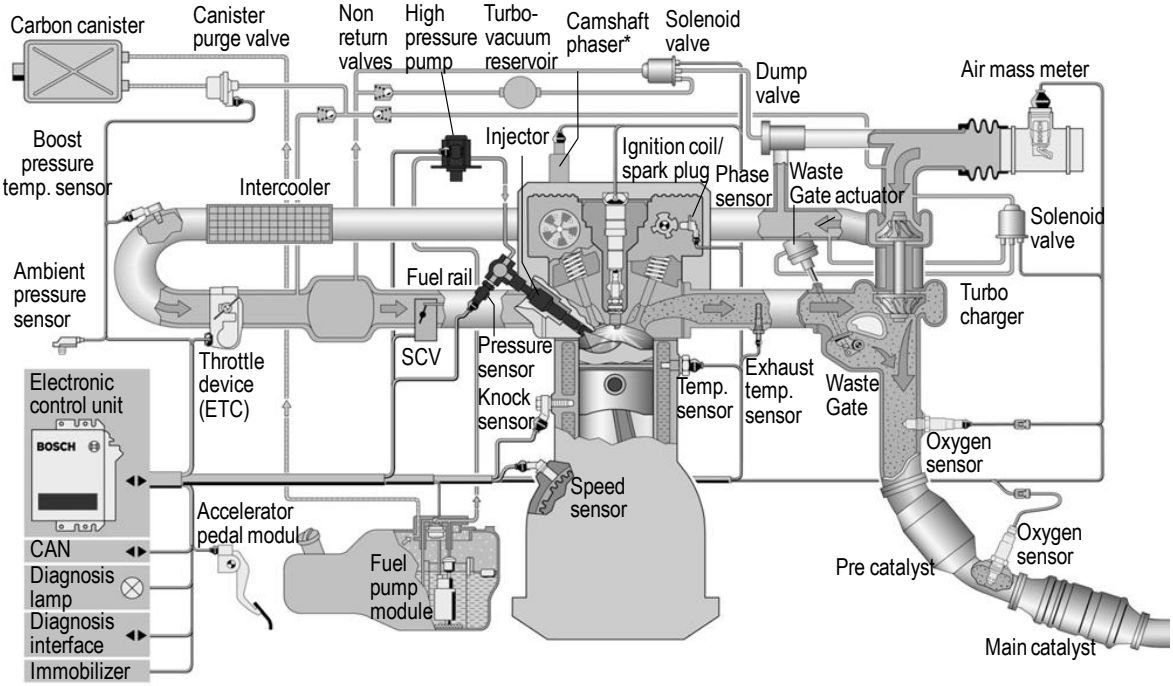


Figure 1.1: Overview of a supercharged gasoline engine with Bosch ECU

To allow optimal operation for the underlying systems and to fulfill legal requirements for diagnosis, the parameters of the ECU functions have to be calibrated and optimised. This is called *ECU calibration*. Caused by the increase of complexity as mentioned at the beginning of this chapter, the calibration process becomes one of the most costly parts of power train development. For optimal engine operation and diagnosis, the ECU implements functions which often include models of the different systems. These models can either be complex physical dynamical models or the usual simplifications, simple maps or curves. In terms of simplicity for implementation and computational effort, multidimensional relations are mostly reduced to two dimen-

sional maps. However, if systems with more than two inputs have to be represented by two-dimensional maps, this approach mostly leads to highly complex structures in ECU software. These solutions can become very hard to implement as well as to calibrate. Fig. 1.2 shows a general overview of the structure between the ECU and an arbitrary dynamical system. In this work, the focus lies on those functions, that either are used for control or for active diagnosis purposes. In both cases, the ECU uses an actuator to impact the dynamical system and receives feedback from a sensor measuring the state of the system. These tasks can become very difficult, especially when sensors are disturbed or corrupted or the system is affected by many other engine variables besides the actual actuator value. It is important to know, that in real-world applications each component will be affected by measurement noise as well as other disturbances during engine operation, which complicates its identification. Different types of inputs in automotive context will be discussed in detail in chapter 2.

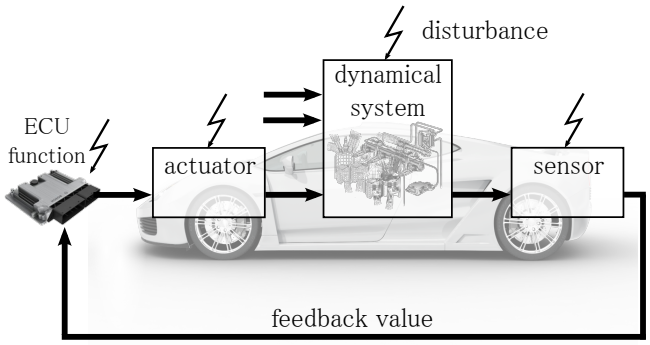


Figure 1.2: Feedback control of a vehicle

However, the increasing complexity of the power train leads to the requirement of a large number of test vehicles and costly engine test benches and roller dyno as well. This prompts the investigation of model-based methods for ECU calibration. An adequate simulation periphery with appropriate plant models delivers a virtual calibration vehicle which responds comparably to a real test vehicle. Thus a transfer of calibration tasks from the real vehicle to the virtual vehicle is pursued. In this context so called *Hardware-in-the-Loop* (HiL) systems seem to be attractive, since these systems are often available in power train development and allow connecting a real ECU directly to a complete vehicle simulation model. Fig. 1.3 shows an overview of a simplified HiL system.

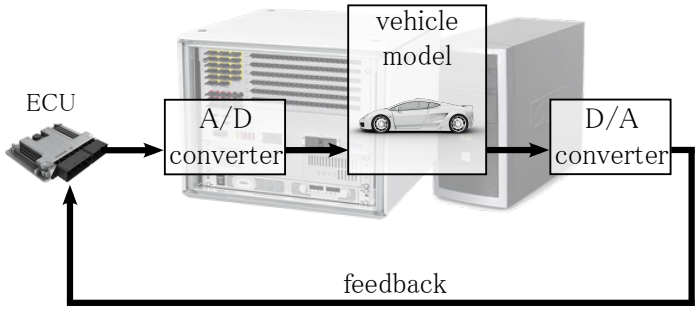


Figure 1.3: HiL as a virtual test vehicle

The virtual vehicle primarily consists of a complete vehicle model including power train, drive train, environment and driver model. A signal box provides D/A and A/D converters to allow communication between the model and the ECU. Compared to a real test vehicle the virtual vehicle has the following advantages:

- High availability
- Controllable environment
- High degree of automation
- Easy to duplicate
- High reproducibility
- Low costs of operation

The disadvantages of virtual vehicles, as compared to real vehicles, also have to be taken into account:

- Simplified reality
- Parametrization costs
- Reduced usage of human senses
- Complicated usability
- Unfamiliarity
- Low fun factor

Virtual vehicles provide a lot of advantages regarding cost efficiency. As a vision of the future, the virtual test vehicle allows a rather decentralised power train development and simplifies vehicle variant development. However, though the concept of virtual vehicles seems to be simple and the aforementioned HiL systems are already state-of-the-art in ECU software development, the difficulty for ECU calibration purposes lies inherently in the problem to efficiently develop such models. As shown in Fig. 1.1, ECU calibration requires a dynamically accurate response of the plant models. Thus, the major task is an efficient development of such models.

To receive highly accurate models incorporating system dynamics and nonlinearities, one can distinguish between theoretical modellig (physical) and empirical modelling (data-based). A comparison of these modelling strategies is given in Fig. 1.4.

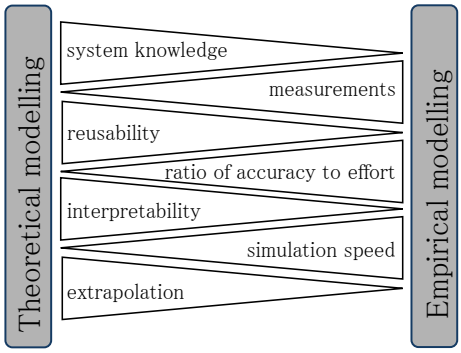


Figure 1.4: Comparison of theoretical modelling and empirical modelling

In general, theoretical modelling requires a deep knowledge of the system and results e.g. in differential equations w.r.t. underlying physics. Theoretical models are interpretable for the user and their parameters can be adapted to further projects e.g. for similar components. Thus, theoretical modelling is well-suited to the general structure of a virtual vehicle though it requires a high investigational effort. Furthermore, if a global vehicle model based on theoretical models exists, it can be reused in all calibration projects by adapting the model parameters to the underlying project specifications. In practice, however, it is a tedious task to develop theoretical models with high accuracy regarding ECU calibration purposes. Either the physics of a system are not fully understood, which leads to wrong model structures, or the theoretical models are too complex to efficiently calibrate them. As a result, particular systems require more efficient methods of identification. Empirical models are well-suited to this requirement. These models need a minimum of system knowledge and deliver highly accurate models and also fast simulation models which can be used for *real-time* applications.
This thesis deals with the identification of nonlinear dynamic systems based on the empirical modelling approach. A consistent tool chain for the identification of dynamic systems will be presented. The principle is shown in Fig. 1.5. Here, a particular system is identified. By dynamically measuring the system inputs and outputs, a dynamical model can be generated using *supervised learning* methods. Then the particular dynamical model is implemented into the complete vehicle model of a HiL System,

based on theoretical modelling. This concept provides highly accurate dynamical models which allow ECU calibration on virtual test vehicles.

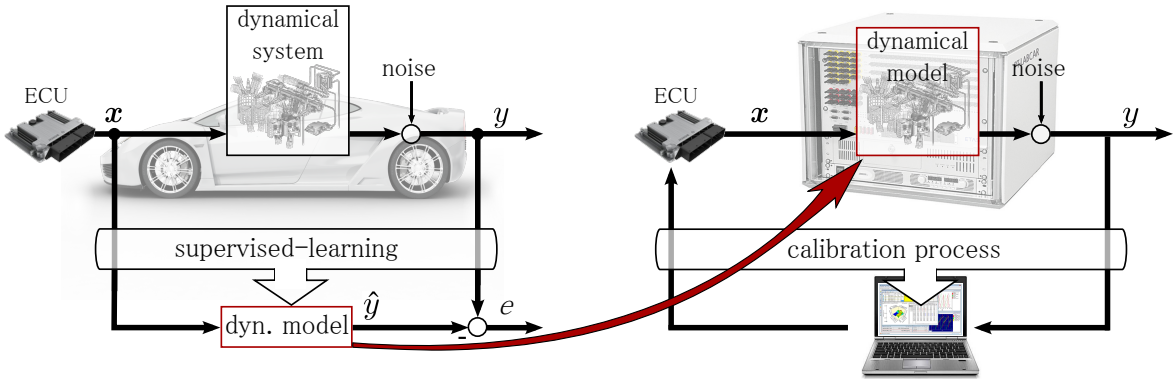


Figure 1.5: Supervised-learning and ECU calibration on a HiL system (Tietze et al. 2014a)

1.2 Related work and contribution

In the last decades a high amount of effort was put into providing physically motivated complete vehicle models for ECU function tests on HiL systems. These models were typically generic and the development was focused on models with a small number of parameters to minimise parametrization effort in practical usage. As only the general functionality of the software is checked during the ECU software test, the function of the models is to let the ECU believe it is in a real vehicle and to keep the reality as simple as possible. There have been analyses of model-based calibration using this simulation environment, see Boumans (2008), Tietze (2011), Çelik (2012) and Raudies (2012). Apart from simple ECU calibration tasks and the potential of doing regression testing on HiL systems, see Tietze & Billand (2012) and Raudies (2012), it has been proven that standard HiL models for ECU software testing do not fulfill accuracy requirements for model-based ECU calibration.

Parallel to HiL investigations, the ECU calibration domain investigated empirical modelling approaches to describe stationary engine behaviour. By starting with simple polynomial models, different model types have been developed for this purpose. Especially Neural Networks (Mitterer 2000, Deflorian et al. 2010), Local Linear Model Trees (Nelles 2001), Volterra Series (Hofmann 2003) and Gaussian Process Regression models (Gutjahr et al. 2011, Berger et al. 2011) were successfully used for mapping

the stationary input/output relationships of an engine, capturing its nonlinearities. These modelling approaches and methods for experimental design, i.e. DoE, are implemented in commercial tools such as ASCMO by ETAS (Kruse et al. 2010, 2007), CAMEO™ by AVL (Bittermann et al. 2004), TOPexpert Suite by FEV (Schlosser et al. 2009) and Easy DoE ToolSuite by IAV (Baumann et al. 2011). Using static nonlinear models for offline applications and especially engine base calibration, e.g. optimizing injection timing and camshaft angles etc., is state-of-the-art, see Mitterer (2000). Today´s investigations to static modelling are related to online applications e.g. Knoedler (2004). The key idea of so-called *Online DoE* is the interaction of the model and engine test bench in order to let the model decide which measurements should improve model quality and to classify which engine states are driveable or not. However, though engine base calibration is an important use case, it is almost the only application for stationary modelling and offline simulation. Implementing stationary engine raw emission models into closed-loop simulation, i.e. the HiL system, was presented in Kruse et al. (2012). Here a combination of physical and dynamical aftertreatment models results in an efficient simulation chain. However, it becomes clear that dynamic models are required for ECU calibration purposes and a dynamical feedback to the ECU is needed/necessary, which is called *closed-loop simulation*. The identification of dynamic systems require new measurement methods. It is commonly called *Dynamic DoE*. Nelles (2001) introduces the Amplitude-modulated Pseudo Random Binary Signals (APRBS) for the identification of dynamic systems. Baumann et al. (2008) used a more applicable chirp signal for excitation because of its smooth transition. Also, excitation of ramps (Godward et al. 2013) or multisine (Tietze et al. 2014a) are in focus for the dynamic DoE. However, if appropriate system measurements are given, a regression algorithm is necessary which can incorporate unknown nonlinearities encoded in the sampled data, and furthermore can model the system dynamics. Investigations of model-based calibration using dynamic data-based modelling are given in Röpke et al. (2012), using Volterra-Series. The dynamic extension for LOLIMOT models is investigated in Hametner & Nebel (2011), Hametner & Jakubek (2012), and also compared to dynamic Multilayer Perceptron Networks (MLP) in Hametner et al. (2013). Dynamical GPR models are also under investigation e.g. Gutjahr (2012). Since one disadvantage of GPR is the cubic computation cost for model training, standard GPR is restricted to a few thousand training points and thus approximation methods are investigated for more efficient modelling (Gutjahr et al. 2013) especially to handle large data sets (Schreiter et al. 2013).
This thesis combines the HiL simulation environment with data-based identification of

dynamical systems. A consistent chain for model-based calibration on HiL Systems is presented. The particular steps of the chain are visualised in Fig. 1.6. Focusing on the identification of nonlinear dynamic systems a data-based modelling approach called Local Gaussian Process Regression (LGPR) (see Nguyen-Tuong et al. (2009, 2008) and Nguyen-Tuong et al. (2010)) will be modified (Tietze et al. 2014b). In Fig. 1.6 the LGPR is visualised as back loops. LGPR allows an iterative and simple modelling procedure and furthermore allows post calibration of dynamic GPR models. The algorithm is able to deal with changing system dynamics and local effects. Moreover LGPR is able to handle large training data sets, by partitioning the training space into smaller submodels.

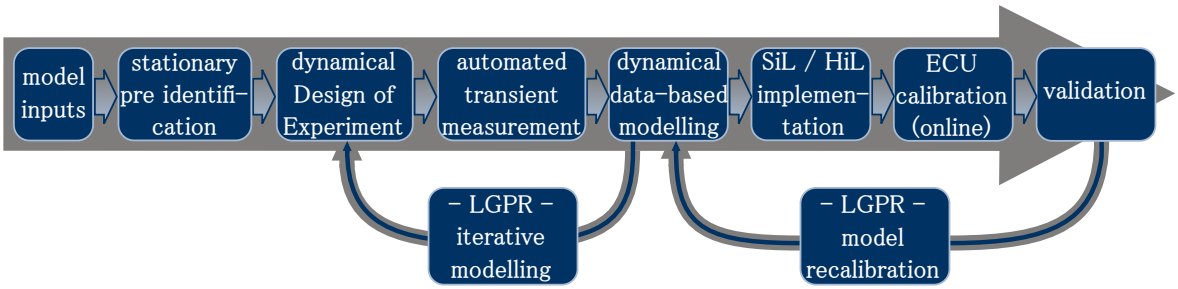


Figure 1.6: ECU calibration using the identification of dynamic systems

1.3 Overview

The themes in this chapter motivate the approach of identifying dynamic systems using data-based methods for ECU calibration purposes. This thesis describes a practical and generic way of identifying dynamic automotive systems as shown in Fig. 1.6. The whole tool chain will be discussed, from the Design of Experiment, modelling of dynamical systems to model-based ECU calibration. All of these particular steps are discussed theoretically and also practically in the last chapter.

Chapter 2 describes the dynamic DoE and gives an overview of system input types for automotive systems in section 2.2. The pros and cons of signal types for excitation will be discussed in section 2.3. It shows how to generate excitation trajectories. In order to ensure safe system excitation a convex hull is used to describe the stationary input signal constraints. Here, the target is to generate appropriate training data for

nonlinear dynamic systems.

Chapter 3 deals with data-based modelling focusing on Gaussian Process Regression. Beginning in section 3.1 an introduction to regression for dynamical nonlinear systems is given and the general requirements for appropriate modelling techniques are discussed. Some important state-of-the-art modelling approaches will be presented and compared to the defined requirements. Neural Networks (section 3.2.2), Neuro-Fuzzy Modelling (Section 3.2.3) and Gaussian Process Regression (section 3.2.4) are briefly presented and analysed.

Subsequently, a comprehensive introduction to the probabilistic theory of the GPR algorithm is given, starting with the Bayesian framework in section 3.3.2 and continuing with the introduction of Gaussian Processes (GP). Finally the powerful regression algorithm of GPR is presented. Since GPR is generally a static modelling approach, the external dynamics structure is therefore applied to the GPR model. GPR shows some drawbacks for modelling dynamic systems of real world applications, since local effects or a change of system dynamics due to engine operation changes, cannot be modelled satisfactorily.

The main contribution of this thesis is the modification of the Local Gaussian Process Regression (LGPR) algorithm (see section 3.4.1) and its application for model-based ECU calibration (see section 4). The concept and the LGPR algorithm is presented in detail by starting with synthetic data to give the reader an intuitive introduction to the principle of LGPR.

After the theoretical parts, Chapter 4 gives the results of the presented algorithm applied to real world examples. Firstly a MATLAB tool is presented, which implies LGPR and provides a graphical user interface (GUI) for easy and intuitive usage. As first application the plant of the high pressure fuel supply control system of a gasoline engine is identified in section 4.2. An example of model-based calibration using frequency response measurement is given in section 4.2.3.

Chapter 5 concludes this thesis with a summary and a brief outlook on future research.

2 Dynamic Design of Experiment

This chapter gives an overview of the Dynamic Design of Experiment (DoE) and points out the difficulty of generating appropriate training data due to the dynamic system measurement. The pros and cons of signal types for excitation will be discussed and how to define the types of system inputs. In order to ensure safe system excitation, a convex hull will be used to describe the stationary input signal constraints and it will be shown how to generate excitation trajectories for safe system excitation.

2.1 Introduction to Dynamic DoE

In the preceding chapter, model-based ECU calibration was motivated using data-driven identification techniques due to the lack of a holistic knowledge of the physical system. As will be seen in chapter 3, the usual task of the modelling step is to identify a latent functional relationship $f(\cdot)$ between a dependent variable y (the system output) and one or more independent variables $x_1 ... x_d$ (the d system inputs) in a continuous manner, based on a set of N observations:

$$y_i = f(\boldsymbol{x_i}) + \varepsilon_i \quad , i = 1, ..., N$$

where ε stands for a normally distributed noise. Since y is measured and can be used for identifying the underlying system, a *supervised-learning* problem is given, see Fig. 1.5 and the step of identifying the unknown function $f(\cdot)$ is called *modelling*. The concept of data-based models is simple and convincing, i.e. measuring a system and quickly generating accurate models without any knowledge of physics (compare Fig. 1.4), but the crucial task in system identification is designing appropriate excitation signals for gathering identification data. Since a trained model can only represent the system behaviour provided by the information within a training data set, a DoE is necessary in the first place for a proper covering or exploration of the system's input

space (Gutjahr 2012). For automotive systems, measuring often requires the usage of test benches or roller dynos which leads to expensive measuring costs, thus the key concept of DoE schemes is to gather as much information about the input-output behaviour within as few measurements as possible (Gutjahr 2012). However, optimal distributed training data strongly depend on the regression algorithm for modelling after the data is collected (Gutjahr 2012) and thus the algorithm should be clear, before generating a DoE plan. This point will be discussed in the course of this chapter. Aside from the regression algorithm, this thesis focuses on dynamic nonlinear systems and thus the intended DoE has to excite the underlying system in such a way that all relevant dynamics and nonlinearities become visible from measured data (Hametner et al. 2013). Generally, DoE methods can be divided into *model-based* and *model-free* approaches (see Deflorian & Klöpper (2009)). If the model and its structure is known, so that a model-based approach is suitable, statistics can be used to define an optimal DoE plan, thus reducing the measuring effort, see Fedorov (1972) and Fedorov & Hackl (1997) for an intensive mathematical introduction. However, in this thesis no prior knowledge of the model is given in most cases so that model-free DoE approaches are used. Furthermore, so-called *nonparametric* models will be introduced in chapter 3, where the distribution of data is modelled rather than assuming a specific model structure (Gutjahr 2012). Basically, model-free DoE leads to *space filling design* (Santner et al. 2003) which generally means to cover the whole input space uniformly by maximization of the minimal distance of the design points to each other (Hametner et al. 2013). Also, in the case of Bayesian regression (in chapter 3), empirical experiments showed that a distribution of the input variables according to a so-called *Sobol sequence* is most suited (see Kruse et al. (2007)). Here, Sobol sequences belong to the class of space filling distributions (see Sobol' (1967)). The property of a Sobol sequence is to maximise the distance between the individual sampling points and to ensure that each sampling value of one dimension occurs only once within the design (Gutjahr 2012). Fig. 2.1 shows a 2D example for the comparison of a random distribution and a space filling design using a Sobol sequence. The space filling design provides a more even coverage of the entire input space due to the distance maximization criterion (Gutjahr 2012).

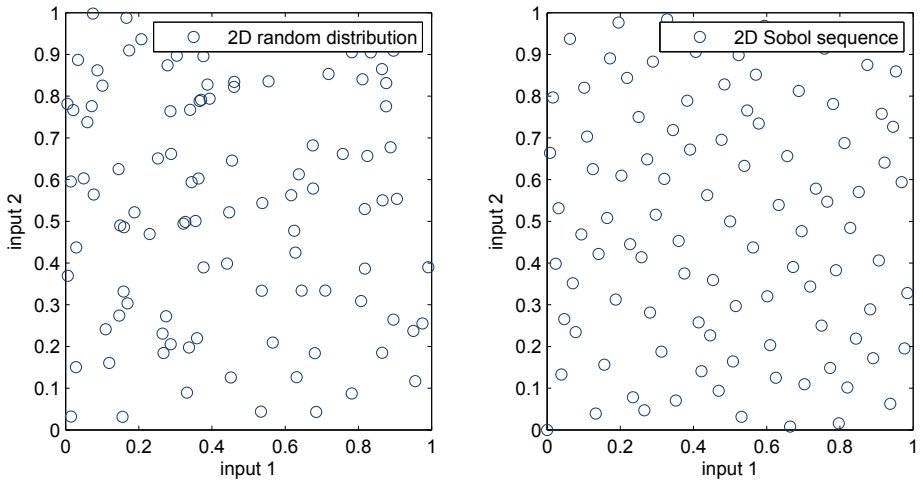


Figure 2.1: Comparison of random distribution (left) and Sobol sequence (right) in 2D

For steady-state applications, space filling DoE was successfully implemented in the aforementioned commercial tools like ASCMO by ETAS, CAMEO™ by AVL, TOP-expert Suite by FEV and Easy DoE ToolSuite by IAV. Here, the user defines the stationary boundaries of the system, which leads to a shrinking of the sample points or cutting the forbidden sample points off. However, regarding dynamic systems, the crucial task is to define dynamic boundaries. Here the main challenge is defining permissible excitation frequencies. The exact dynamic boundaries are mostly not known beforehand and thus need to be estimated. Fig. 2.2 shows the principle of different types of system hulls for a two-dimensional input space. The green crosses describe the measurement of the stationary points. When dealing with high dimensions the resulting hull is concave and it is not easy to deal with it in a mathematical way. It is therefore useful to use the convex, and further a conservative convex hull in the first identification step. After generating the convex hull of the input space, appropriate system excitation has to be imposed on the system to receive the dynamic responses.

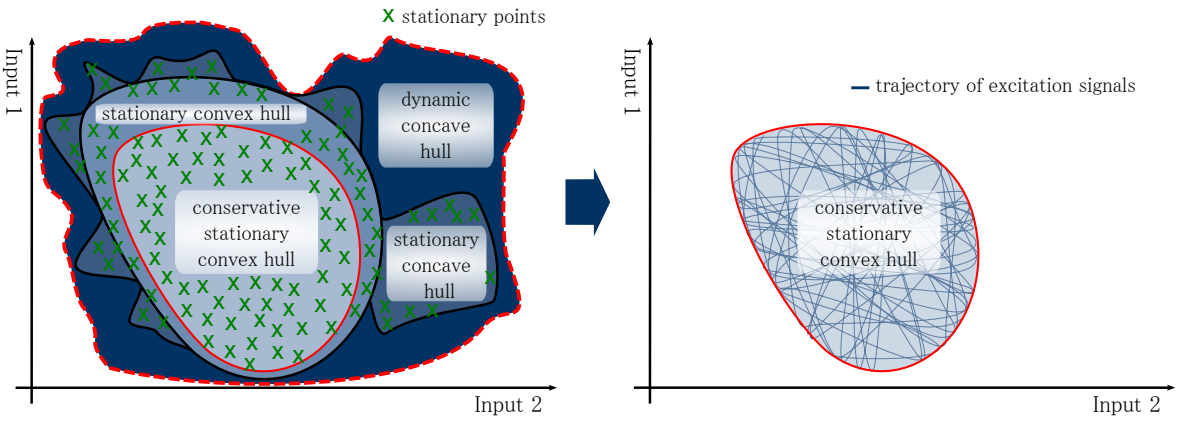


Figure 2.2: Types of system hulls (Tietze et al. 2014a)

However, to receive a globally valid model of the system, the main task of the dynamic DoE is to find the dynamic boundaries in which the system can be excited and furthermore to design appropriate excitation signals in order to fulfill the space filling requirement. To summarise, the theoretical steps of the dynamic DoE part are given as:

- *classification of system inputs and outputs*,
- *using well-suited excitation signals*,
- *defining dynamic boundaries*, and
- *confines the excitation signals within the safe input space.*

In the following sections, each step will be discussed in detail.

2.2 Input/output classification

Given Fig. 1.6, the first step of identifying dynamic systems is to identify all system inputs and outputs and finding the relevant inputs with respect to excitation, system protection and usage of the model. An overview of system inputs in automotive context was presented in Tietze et al. (2014a).
Dealing with mechanical processes the influence of the different variables is usually quite clear (Nelles 2001). Although the inputs of the system are known they differ in their properties for identification. Fig. 2.3 shows the decomposition of an automotive

system. It is divided into the dynamical system, which has to be identified, the vehicle itself, the ECU and the environment.

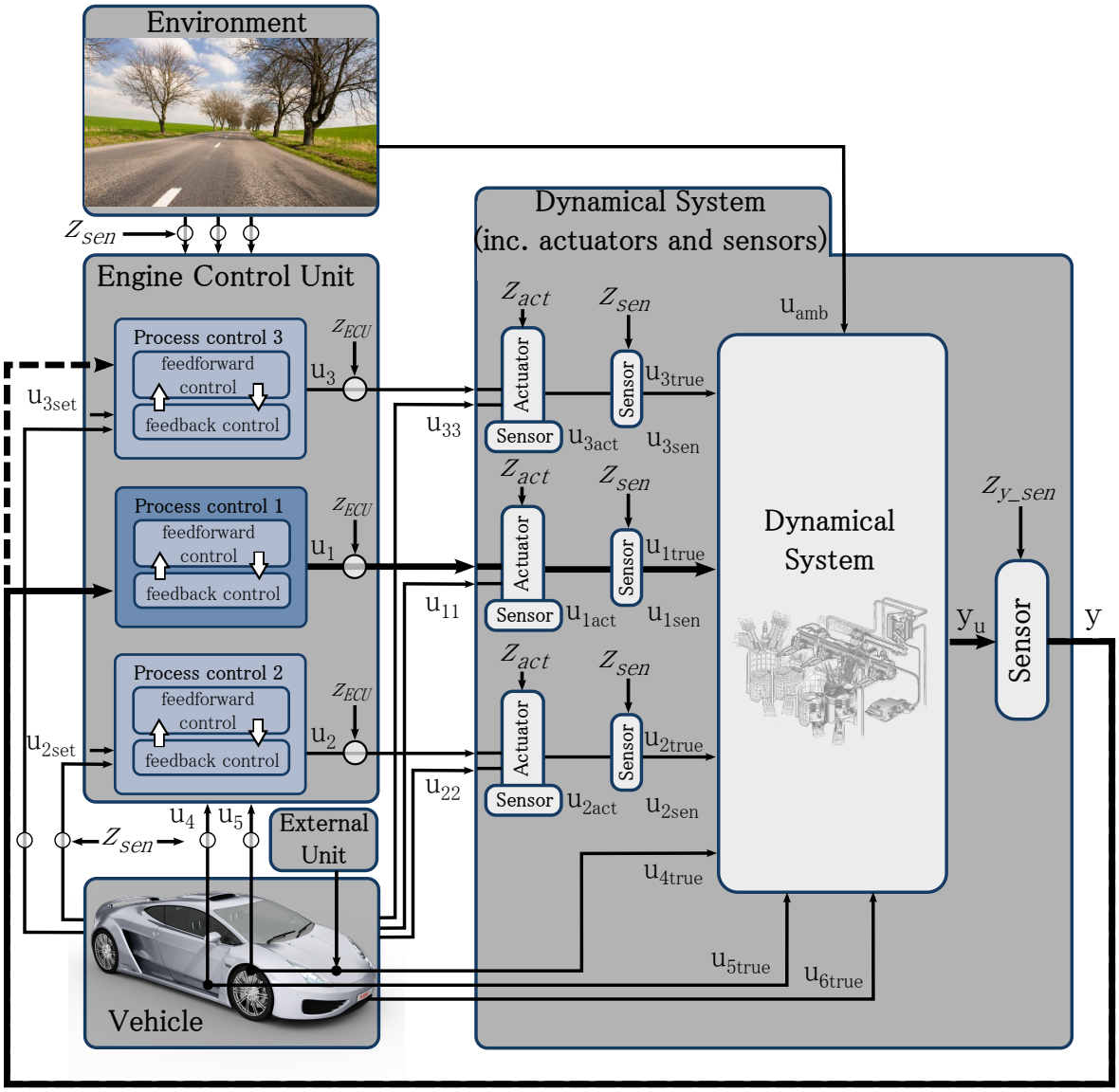


Figure 2.3: Different types of system inputs (Tietze et al. 2014a)

In this thesis, six different types of inputs are distinguished. Type 1 inputs are most suited for identification purposes whereas type 6 inputs are most difficult to deal with.

- *Type 1 input (*$\mathbf{u_1}$, $\mathbf{u_{11}}$, $\mathbf{u_{1true}}$, $\mathbf{u_{1act}}$ *and* $\mathbf{u_{1sen}}$*):* Generally, the controlled system output y is mainly affected by the control signal u_1 which can directly and easily be actuated by the ECU and is therefore well-suited for identification.

In fact u_1 is not a direct input of the system itself, since the actuators and disturbances affect u_1. Although u_{1true} would be the best input for identifying the underlying system, it is not possible to get the true value and thus either the sensor value u_{1sen} of u_{1true} or the sensor value of the actuator position u_{1act} can also be used as input, if sensors with good dynamical properties are installed. In fact, the free excitation of u_{1act} and u_{1sen} is not feasible in general. Since the goal of model-based controller calibration is to control y as good as the controller structure allows and the only possible interface of the controller is u_1, it is useful to take the actuator as part of the dynamical system. Notice that u_1 is typically a control signal and thus, when taking u_1 instead of a physical value (u_{1act} or u_{1sen}) as system input, the mechanical input u_{11} of the actuator, e.g. hydraulic pressure, also has to be taken into account.

- *Type 2 input (*$\mathbf{u_2}$*,* $\mathbf{u_{22}}$*,* $\mathbf{u_{2true}}$*,* $\mathbf{u_{2set}}$*,* $\mathbf{u_{2act}}$ *and* $\mathbf{u_{2sen}}$*):* Type 2 input has the same configuration as type 1 input, but u_2 is the output of the system control of another dynamical system. It also affects the dynamical system which has to be identified. Since this controller is already calibrated, it is possible to use further types of inputs. Using u_{2set} as an input for identification would lead to the ECU software becoming part of the system itself. Because of software structures like state machine and hysteresis, it is not advisable to make the freecut in front of the ECU software. However, using the set point u_{2set} for excitation and the sensor of either the actuator u_{2act} or the sensor value u_{2sen} as the system input is an attractive option. The advantage of using u_{2set} for excitation is that it is a more secure operation since the controller restricts the dynamics of the excitation and thus prevents the system from being destroyed.

- *Type 3 input (*$\mathbf{u_3}$*,* $\mathbf{u_{33}}$*,* $\mathbf{u_{3true}}$*,* $\mathbf{u_{3set}}$*,* $\mathbf{u_{3act}}$ *and* $\mathbf{u_{3sen}}$*):* Type 3 input is nearly the same as type 2 input except that the feedforward control is affected by the system output y. If it is too risky or impossible to open this loop, the challenge is to identify the system in closed loop (Isermann 2011) and thus it becomes more difficult. Note that if y is part of the feedforward control 3 it is possible that y is the mechanical input of the actuator u_{33}. This means that for the structure of the dynamical model it is important to use the system output also as a delayed model input.

- *Type 4 input (*$\mathbf{u_4}$*,* $\mathbf{u_{4true}}$*,* $\mathbf{u_5}$ *and* $\mathbf{u_{5true}}$*):* Not all inputs of the dynamical system can be actuated by the ECU. In fact the true values u_{4true} and u_{5true} of u_4 and u_5

cannot be measured by the ECU since the filtering effects of the sensors have to be taken into account. Here, with respect to the identification of the dynamics, it is important to use sensors with known tolerances, small time constants and small time delays. Furthermore, it is advisable to use so-called mean value sensors if just one system is available for measurements. However, since the actuation of u_4 and u_5 is not directly possible and thus costly to realise, it is important to analyse whether the input can be seen as a disturbance or must be taken as a real input. In Figure 2.3 the input u_5 is assumed to be real input and thus has to be excited by an external unit. The input u_4 can either be used as input or as disturbance, depending on the strength of its effect on the system.

- *Type 5 input (*$\mathbf{u_{6true}}$*):* u_{6true} is not measurable by the ECU and thus also not measurable in the simulation environment. It is advisable to define u_{6true} as a disturbance. If u_{6true} strongly affects the system it can be transformed into an input of type 4 by installing an additional sensor. However, this procedure is costly since the additional sensors should be available during the whole calibration phase as this input requires further models to estimate the sensor value of u_{6true}.

- *Type 6 input (*$\mathbf{u_{amb}}$*):* $\mathbf{u_{amb}}$ describes different environmental conditions like ambient temperature, air pressure, air humidity or solar radiation. In fact, the excitation of these values is quite difficult and it is therefore always a good approach to treat the environmental conditions as a disturbance.

Input selection is a key step since it influences all the following steps of the identification process chain. Due to the second step, DoE, the number of inputs increases the measurement time exponentially and similarly increases the measurement costs. It is therefore advisable to reduce the number of inputs to a minimum. In addition, regarding the usage of the resulting model for simulation in closed loop, a simpler model is more robust than a model with many inputs, for instance, the extrapolation behaviour is less crucial. In most cases a trial and error principle is used to identify the relevant inputs for a stationary problem. However, it will be shown in the next section, that next to the physical inputs, the number of inputs further increase by the dynamical modelling structure. Thus the trial and error principle can be a tedious task. Using so-called *Feature Selection* methods can help to identify the relevant inputs (see Markert et al. (2011)) for a comparison of different Feature Selection methods).

2.3 Requirements for excitation signals

Given the relevant system inputs, the task of the DoE step is to generate data, incorporating all relevant dynamics and nonlinearities. The evaluation of appropriate data is related to the underlying dynamics modelling approach. Again as will be seen in chapter 3, the most frequently used approach for data-based modelling of dynamical systems is a combination of a *static nonlinear approximater* with an *external dynamic filter bank*, see Nelles (2001). Fig. 2.4 gives an overview of the external dynamic structure. Here, the input u and the output y are time delayed before entering the static approximator. k denotes the time step.

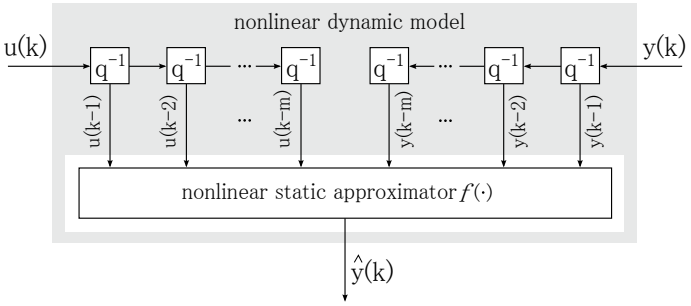


Figure 2.4: External dynamics structure (Nelles 2001)

Regarding the structure of this dynamical model the following properties for the excitation arise:

- The input space is extended by the delayed inputs and thus increases strongly with the order of the system.
- The approximator inputs cannot all be influenced directly and independently. Rather, only $u(k)$ is chosen by the user, and all other delayed approximator inputs and outputs follow as a consequence (Nelles 2001).
- The lower the frequency of the input signal the closer the data will be to the static nonlinearity (equilibrium) of the system (Nelles 2001).
- Naturally, the data distribution is denser close to the static nonlinearity than it is in off-equilibrium regions, since systems with autoregressive components approach their equilibrium infinitely slowly (Nelles 2001) (see Fig. 2.5).

- Highly dynamic input excitation is required in order to cover wide regions of the input space with data.

- Different combinations of frequencies can excite different dynamic modes of the model. (Tietze et al. 2014a).

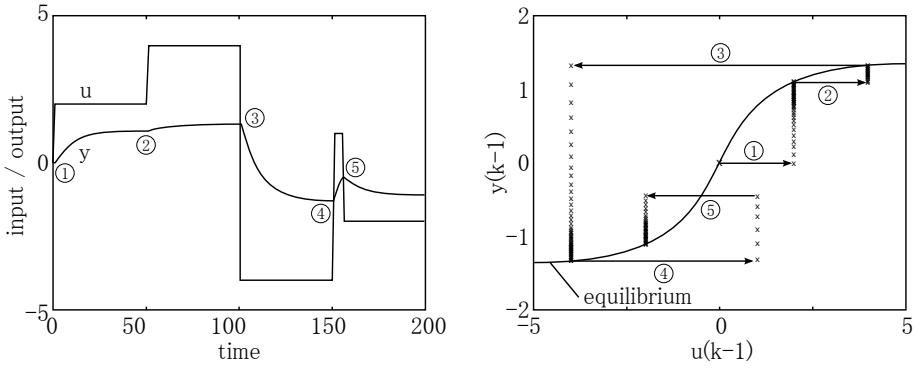


Figure 2.5: Correspondence between the input/output signals of a system (left) and the input space of the approximator in external dynamic approaches (right) for a step-like excitation (Nelles 2001)

Given the model structure, and regarding the main requirement of *space filling property*, the requirements of nonlinear dynamics system excitation can be defined:

- *Uncorrelated excitation*: In order to allow the modelling algorithm to discriminate between the individual effects of the various inputs to the system response, the individual designs for a multiple-input system have to be uncorrelated (Gutjahr 2012). The similarity of two time series can be computed by their *cross correlation*, for time-discrete signals given as:

$$R_{x_1x_2}(\tau) = \sum_{k=1}^{N} x_1(k)x_2(k+\tau)$$

  Here, the index k is taken to be the time and τ the time *lag*. Fig. 2.6 shows an example of correlated input signals. Here, two noisy cosine functions of different frequency show a high correlation and would not fulfill the requirement of uncorrelated excitation.

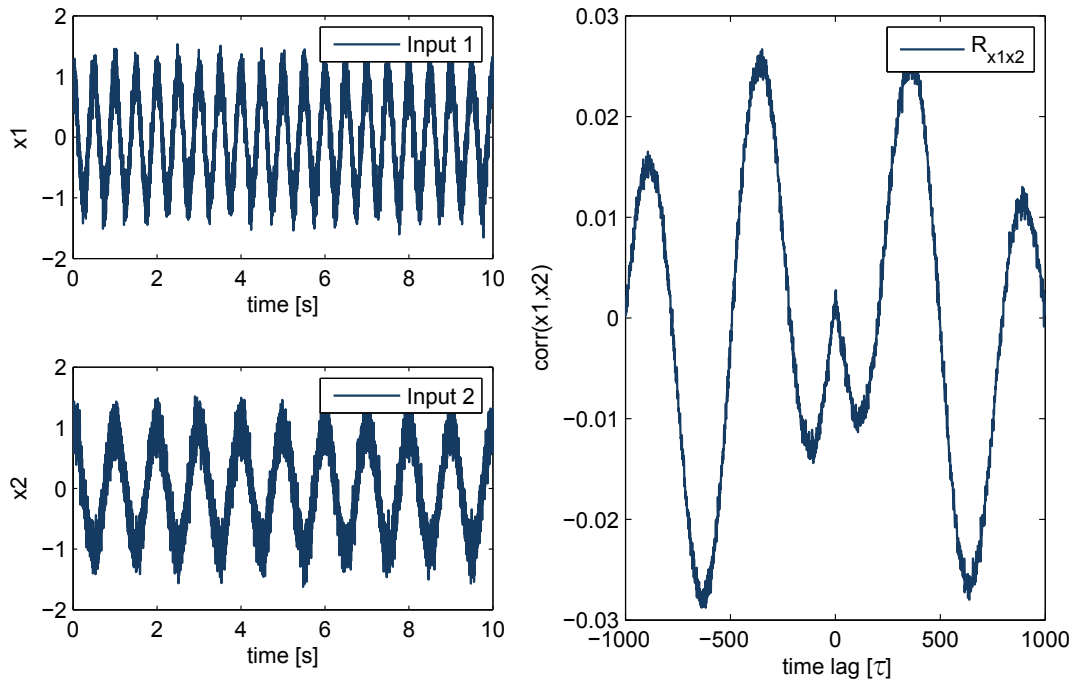


Figure 2.6: Correlation of two noisy cosine functions

Another statistical measure for correlation is given by *Pearson's correlation coefficient* given as the covariance of the two variables divided by the product of their standard deviations:

$$\rho = \frac{cov(x_1, x_2)}{\sigma_{x_1}\sigma_{x_2}} = \frac{E[(x_1 - \mu_{x_1})(x_2 - \mu_{x_2})]}{\sigma_{x_1}\sigma_{x_2}}$$

The sample Pearson's correlation coefficient then is given as:

$$r_{x_1,x_2} = \frac{\sum_{i=1}^{n}(x_{1i} - \bar{x}_1)(x_{2i} - \bar{x}_2)}{\sqrt{\sum_{i=1}^{n}(x_{1i} - \bar{x}_1)^2}\sqrt{\sum_{i=1}^{n}(x_{2i} - \bar{x}_2)^2}}$$

- *Selectable frequencies*: As mentioned, the main task of dynamic DoE is to find the dynamic boundaries. This means the permitted frequencies and their amplitudes. Vice versa, frequencies exist which are not permitted and the excitation with these frequencies would lead to the system being destroyed. Constrained frequencies further lead to *smooth signal transition* and thus ensure a safe excitation.

- *Selectable amplitudes*: Given the permitted frequencies, the amplitudes must also be constrained as the dynamical boundaries are influenced by frequency and amplitude.

- *Amplitude Spectrum*: In order to evaluate an excitation signal regarding frequencies and corresponding amplitudes, the periodic signal can be analysed in frequency-domain and therefore has to be transformed by using *Discrete Fourier Transformation*, see appendix A.2. The spectrum of the signal can be seen as dynamic input space. In order to achieve a dynamic input space filling, the spectrum of the signal should cover the relevant frequencies. To achieve a good coverage of the spectrum, the property of *linearity* of the Fourier Transformation can help to design appropriate excitation signals. Fig. 2.7 illustrates the property of *linearity* of the Fourier Transformation. In the upper two plots, two sinusoids and their amplitude spectra are given. The sum of both signals leads to the amplitude spectrum which consists of both frequencies with the same amplitudes. Thus, for instance, the coverage of the amplitude spectrum can be reached by adding periodic signals (note that generally the sum of two periodic signals is not periodic anymore. In order to receive periodic signals, the ratio of the frequencies must be a rational number). However, it can be seen that the amplitudes of the signal $x_1 + x_2$ are higher, which can be critical for excitation. Normalising the signal to the given boundary would lead to smaller amplitudes (see the lower plot). To counteract this dilemma, the so-called *Crest factor* will be introduced as the next requirement for excitation signals.

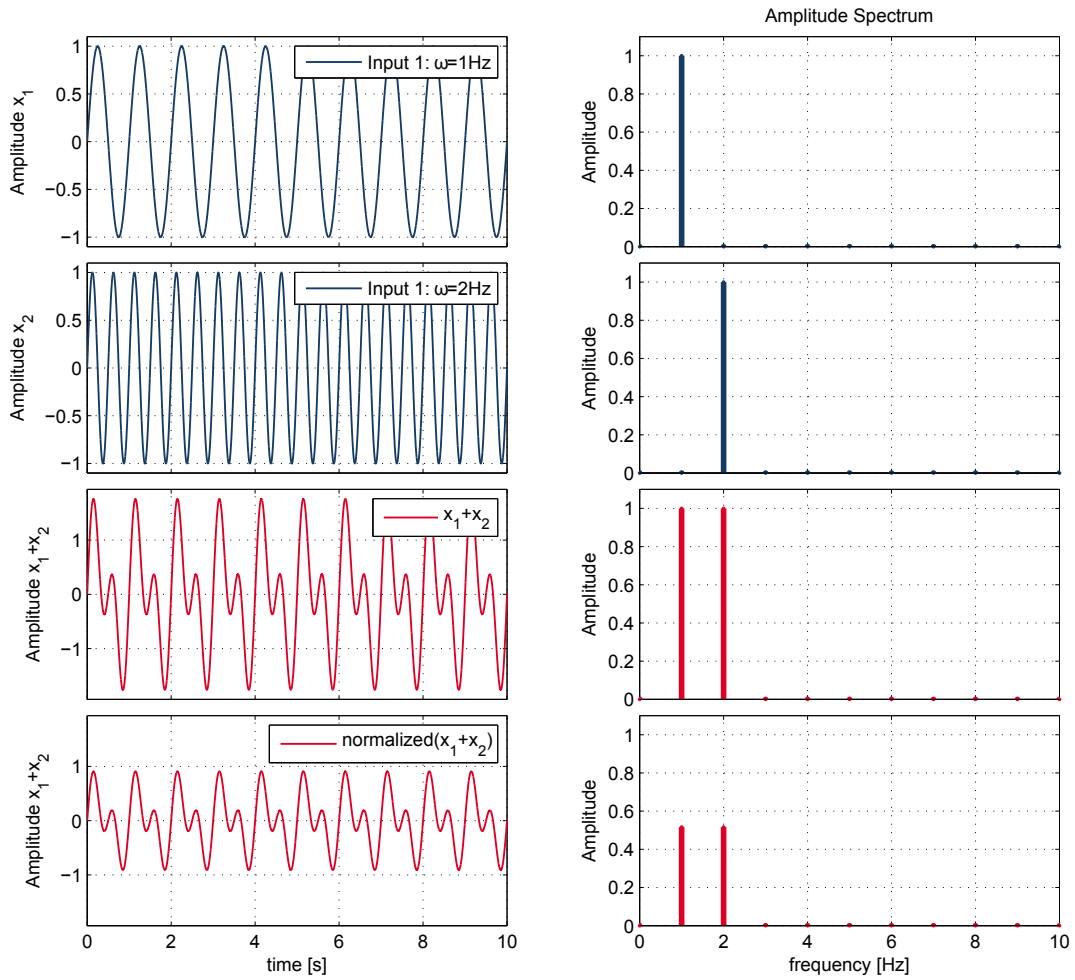


Figure 2.7: Linearity of Fourier Transformation

To generate excitation signals with appropriate spectra, the generation of the signal can be done in the frequency-domain. To that purpose the *convolution theorem* can be used (see appendix A.2). By sampling in the frequency-domain at multiples of $1/T$, which can be described as a multiplication with a *Dirac train* (see Eq. A.4.1.1) so that in the time-domain a convolution should be made with a Dirac train $T\delta_T(t)$ (Pintelon & Schoukens 2012).

- *Crest factor*: To get an idea of the compactness of a signal, the crest factor $Cr(u)$ of a signal $u(t)$ is given by the ratio of the peak value u_{peak} of the signal to its *rms* value u_{rms} in the frequency band of interest (Pintelon & Schoukens 2012).

$$Cr(u) = \frac{u_{peak}}{u_{rms}} = \frac{\max|u(t)|}{\sqrt{\frac{1}{T}\int_0^T u^2(t)dt}}$$

with T as the measurement time. In order to inject a lot of power into the system, the crest factor of the signal should be small. Taking the example of Fig. 2.7 the sum of x_1 and x_2 can be *crest factor optimised* by varying the phase of the sinusoids. Fig 2.8 shows the result of the standard sum of signals x_1 and x_2 and the crest factor optimised result in the lower plot. Although both maximal amplitudes of the global signal are limited to values between -1 and 1, the crest factor optimised signal reaches higher amplitudes for the frequencies of 1Hz and 2Hz.

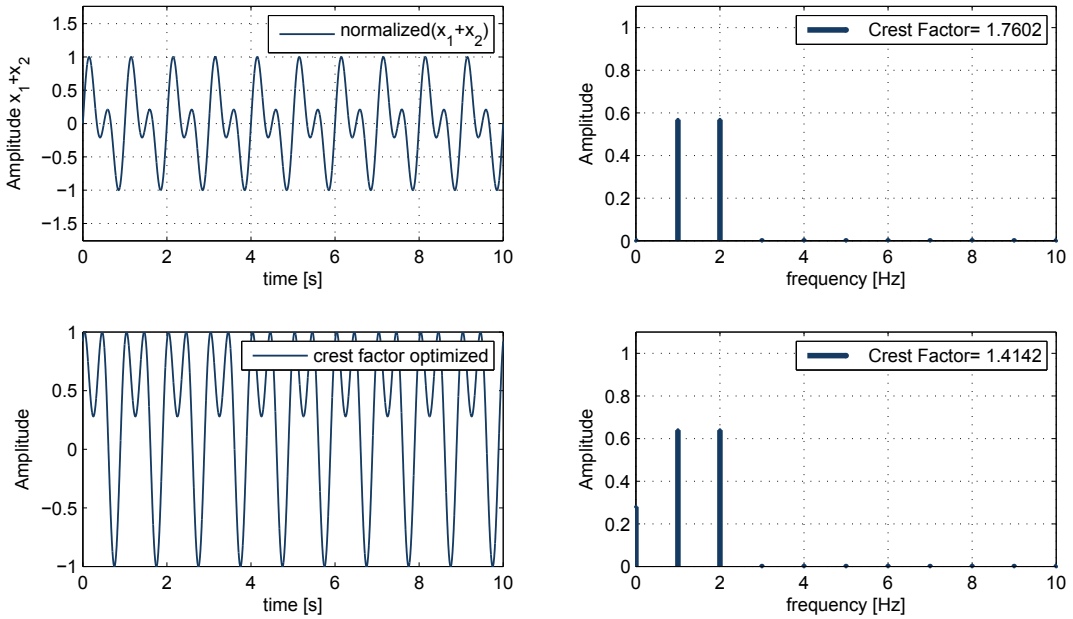


Figure 2.8: Crest factor optimised sum of sinusoids

- *Frequency combination*: Since a typical system is of the type *multiple input single output* (MISO), the dynamical boundaries are given as the combination of input excitation frequencies. Thus it is not sufficient to design a dynamic DoE regarding the frequency spectra of each input signal individually. The excitation signals must ensure that all excited frequencies are combined with all excited frequencies of the other inputs. Note that the phase of the signals is another degree of freedom and has to be shifted in order to reach a space filling design.

- *Amplitude combination*: As the system is nonlinear, the requirement of frequency combinations has to be extended to amplitude combinations. Thus all relevant frequencies of each input have to be combined with all relevant frequencies of the other inputs, shifted phases have to be ensured and variations of the amplitudes

have to be realised. The input space for two periodic input signals consists of at least two frequencies, two amplitudes and one phase shift. Thus the input space scales with the dimensionality $(D \times 3) - 1$.

- *Excitation duration*: The input space increases exponentially with the number of inputs. For instance, given a function with three inputs, for which 125 samples are necessary for identification, the corresponding dynamical input space leads to nine dimensions and thus requires approximately two million sample points in order to achieve the equivalent design space coverage. However, the high dimensionality leads to a huge input space in which samples must be generated very efficiently.

- *Space filling distribution*: To fulfill the space filling property efficiently, the distribution of the data in the input space, regarding each input dimension must be taken into account. Thus a uniform distribution should be pursued. For instance, Fig. 2.9 shows two multisine signals and their distributions.

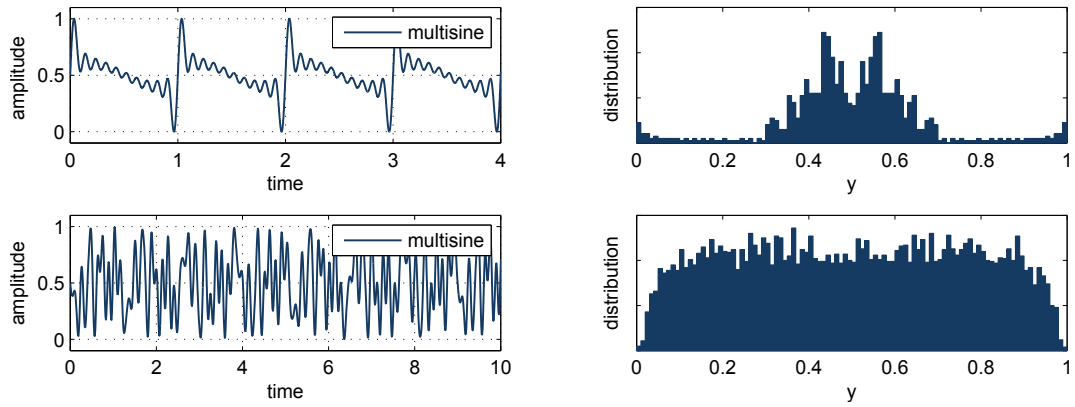


Figure 2.9: Example of two excitation signals with bad space filling distribution (upper plot) and good space filling distribution (lower plot)

All of these criteria can be used to design appropriate excitation signals to fulfill the requirement of space filling data distribution. In the following section, different commonly known excitation signals will be presented and analysed with respect to the requirements.

2.4 Comparison of excitation signals

The last section introduced criteria in order to create appropriate excitation signals for nonlinear dynamic system identification. In this section, the commonly known excitation signals will be discussed:

- *Amplitude modulated Pseudo Random Binary Sequence*
- *Ramps (Ramp and hold)*
- *Multisine (Crest Factor optimised)*
- *Shifted Chirps*

2.4.1 Amplitude modulated Pseudo Random Binary Sequence

For linear systems, a suitable excitation signal is the so-called *pseudo random binary sequence* (PRBS), see e.g. Isermann (2011). Since in contrast to linear systems, for nonlinear systems the property of *superposition* is not given and therefore the amplitudes of the excitation signals have to be varied. An example of why a PRBS signal is not appropriate for nonlinear system identification is given in Nelles (2001). An extension to the PRBS signal is to give different amplitudes to each step of the PRBS signal. This is called *Amplitude modulated Pseudo Random Binary Sequence* (APRBS), see Nelles (2001). Aside from the minimal and maximum amplitudes and the length of the signal, the *minimum hold time* T_h is a further design parameter. Given the length of the signal, the minimum hold time determines the number of steps in the signal and thus influences the frequency characteristics (Deflorian & Klöpper 2009).

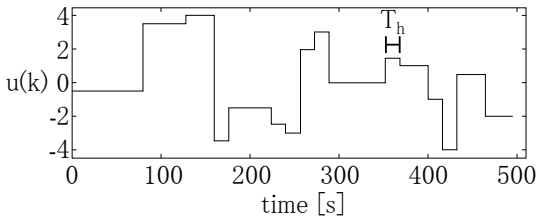


Figure 2.10: APRBS signal with minimum hold time T_h (Nelles 2001).

Initially, the APRBS amplitudes were chosen randomly. Thus for high dimensional

inputs the input space develops holes. To obtain an optimal DoE, the APRBS signal must be designed to fulfill the property of space filling. For instance, Deflorian & Klöpper (2009) presented a *Maximin Latin hypercube APRBS design* which guarantees a better distribution of the design points.

APRBS for identification of dynamic nonlinear systems: Many papers in the field of DoE describe the APRBS signal as a suitable excitation signal for the identification of dynamic systems, see Deflorian & Klöpper (2009), Deflorian & Zaglauer (2011), Schreiber et al. (2011). APRBSs are simple to generate and often the excitation is simple to transfer to the real system.
Although the APRBS signal excites all frequencies, it is important to note that the amplitudes decrease with increasing frequencies. This can be shown by the approximation of the step signal with a Fourier Series and thus by the spectrum in frequency-domain (see Fig. 2.11 right sub plot). Also, in time-domain the disadvantages of APRBS signals can be shown by plotting the stationary (Fig. 2.11 left sub plot) and the dynamical input space here for simplicity for one input signal (Fig. 2.11 middle sub plot). Generally, the stationary input space is well covered, but the orthogonal directions of the APRBS signal lead to a sparse coverage of the dynamical input space. Also, a practical disadvantage due to the non-selectable excitation frequencies is given. Thus the step excitation is the major drawback regarding safe excitation. This is why APRBS signals often cannot be used for practical identification applications.

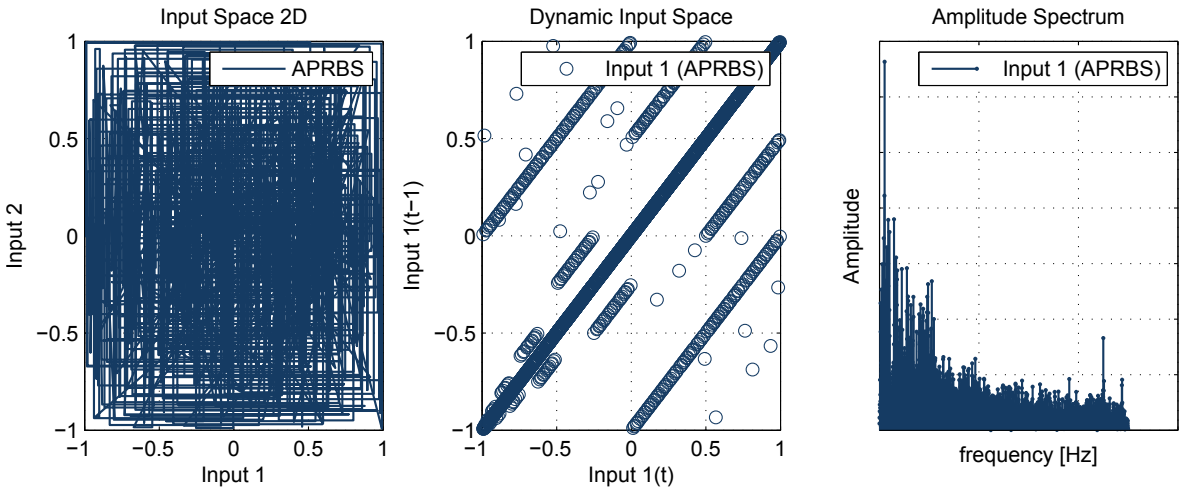


Figure 2.11: APRBS: input space (left), dynamic input space (middle) and spectrum (right)

2.4.2 Ramp Signals

Generally, Ramp signals are very similar to APRBS signals. Instead of a step, the amplitude is reached by a ramp (see Fig. 2.12).

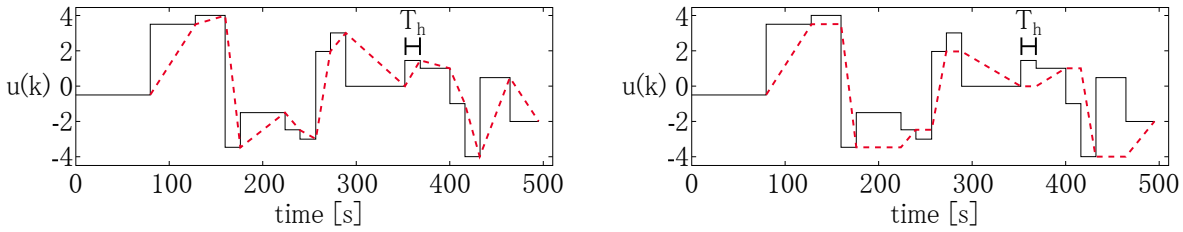


Figure 2.12: Ramp signal (left) and Ramp-and-hold signal (right)

The Ramp signal can also be generated with hold phases (see Godward et al. (2013) for instance). Furthermore, a Sobol distribution of amplitudes and slopes leads to a good coverage of multidimensional DoE plans.

Ramp signals for identification of dynamic nonlinear systems: The input space distribution of the Ramp signal is concentrated at the centre of the input space (see Fig. 2.13, left sub plot). The dynamic space shows a better coverage compared to the APRBS signal and the spectra are very similar. The Ramps lead to a stronger decrease of the amplitudes for increasing frequencies. Furthermore, the Ramp Signal has a worse Crest factor compared to the APRBS signal.

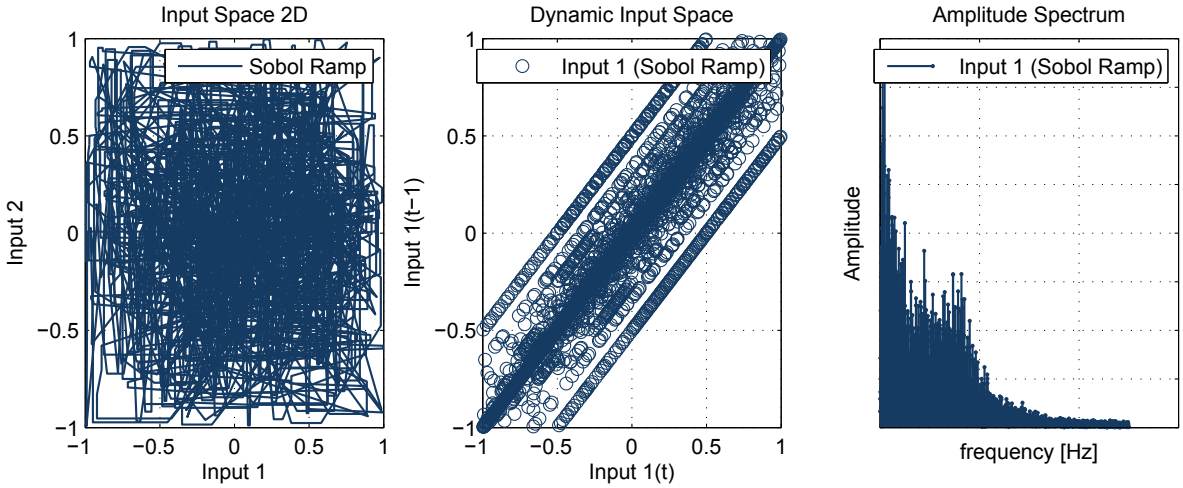


Figure 2.13: Sobol Ramp: input space (left), dynamic input space (middle) and spectrum (right)

To summarise, the Ramp signal analysis shows benefits concerning the selectable frequencies which are more practical for the excitation of real dynamic systems compared to the APRBS signal. Furthermore, the input space shows a better coverage.

2.4.3 Multisine (Crest Factor optimised)

A multisine is a typical excitation sequence for nonparametric system identification using frequency response measurement (see Pintelon & Schoukens (2012)). It is a sum of F harmonically related sine waves, given as:

$$u(t) = \sum_{k=1}^{F} A cos(2\pi f_k t + \phi_k)$$

with phases $\phi_k = -k(k-1)\pi/F$ and $f_k = l_k f_0$ with $l_k \in \mathbb{N}$. In order to improve the multisine signal, the phase relations are optimised by a numerical search method. The literature provides two methods to optimise the *Crest factor*, a *clipping procedure* that cuts the largest peaks of the signal, or the so-called *Infinity Norm Algorithm*, see Pintelon & Schoukens (2012) for a discussion of both approaches. By designing a multisine for multi input excitation, the requirement of uncorrelation has to be taken into account (see section 2.3).

Multisine signals for identification of dynamic nonlinear systems: The multisine allows a free selection of the frequencies on the discrete grid $l_k f_0$. Also, the amplitudes of the harmonic components can be chosen freely. It is guaranteed that no out-of-band power appears and the crest factor is small. The coverage of the input space is very good, see Fig. 2.14, left plot and the dynamical example for one input in the middle plot.

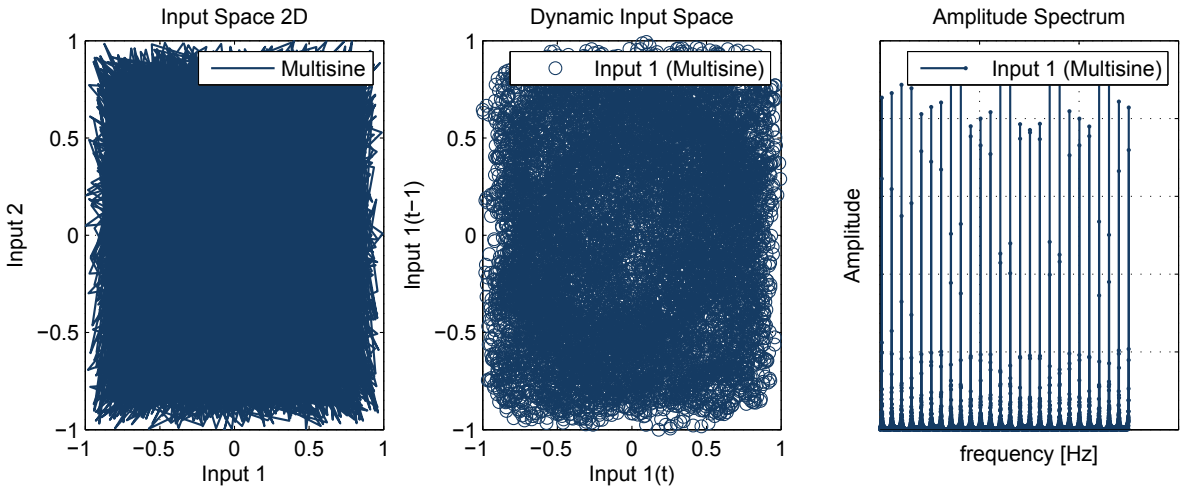


Figure 2.14: Multisine: input space (left), dynamic input space (middle) and spectrum (right)

2.4.4 Shifted Chirp

A swept sine or periodic chirp is a sine sweep test where the frequency is swept up and/or down in one measurement period:

$$u(t) = Asin((at + b)t) \quad 0 <= t < T_0$$

with T_0 the period, $a = \pi(k_2 - k_1)f_0^2$, $b = 2\pi k_1 f_0$, $f_0 = 1/T_0$, $k_2 > k_1 \in \mathbb{N}$ and $k_1 f_0, k_2 f_0$ the lowest and the highest frequency (Pintelon & Schoukens 2012). A visualisation of the chirp signal is given in Fig. 2.15.

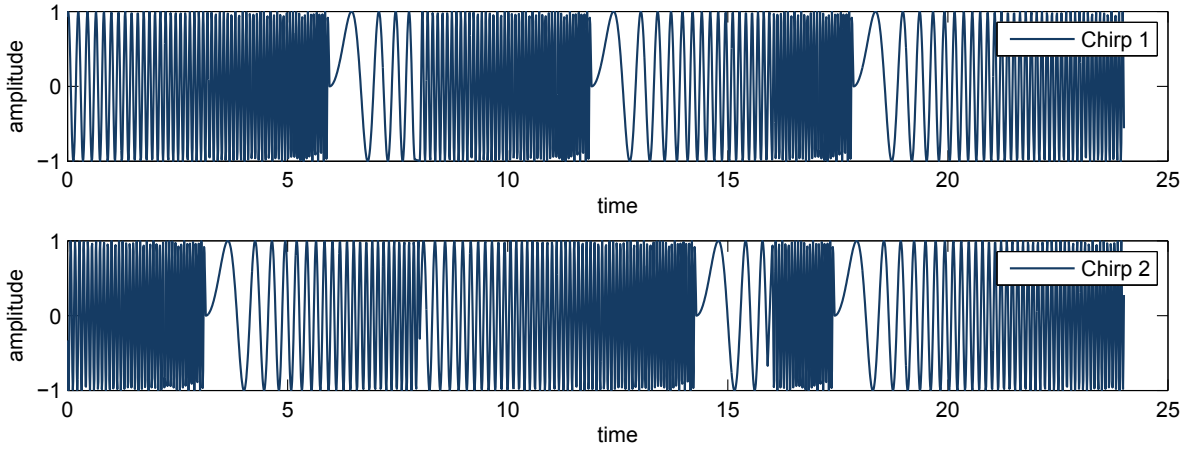


Figure 2.15: Chirp signal

It can be seen that the two input chirps are shifted in order to get a good coverage of the input space. The method of shifting can be found in Nguyen-Tuong, Markert & Meister (2014).

Shifted chirps for identification of dynamic nonlinear systems: An advantage of the Chirp signal compared to the APRBS signal and the Ramp signal is that most of the power is equally distributed in the user-selected frequency band $[k_1, k_2]f_0$ with $k_2 > k_1 \in \mathbb{N}$ (Pintelon & Schoukens 2012). This can be seen in the frequency domain, see Fig. 2.16 right plot. The input space is well covered extending to the boundaries (see Fig. 2.16 left plot). Also, the dynamic input space (here for simplicity shown for one input) is well covered by two shifted chirp signals (see Fig. 2.16 middle plot).

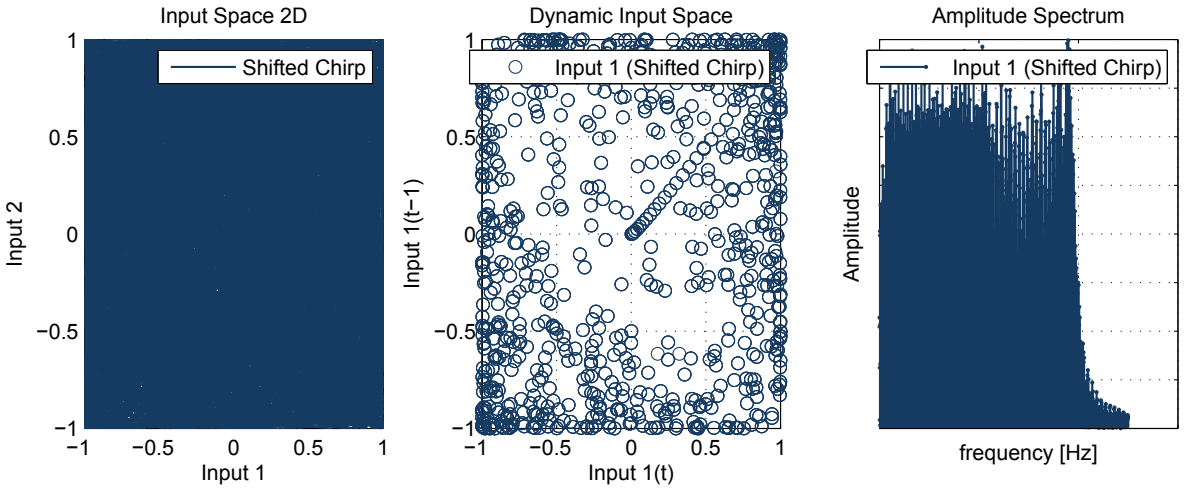


Figure 2.16: Shifted Chirp: stationary Input space (left), dynamic input space (middle) and spectrum (right)

The shift of the chirp is necessary in order to guarantee that all frequency combinations across the inputs can be obtained, thus the chirps have to be shifted sufficiently many times. Chirp signals have a small Crest factor (typically 1.45) and a smooth transition, which gives a good applicability for the excitation of automotive systems.
The main disadvantage of Chirp signals is the long measurement time that is required in order to cover the whole input space. Obviously this disadvantage strongly increases with the number of input dimensions.

2.5 Identification of system boundaries

Before the experiment can be designed, it is necessary to get information about system boundaries and to estimate how the system reacts when passing these boundaries. As shown in Fig. 2.2, it is important to distinguish between the *stationary input space* and the *dynamical input space*. Starting without prior information about the underlying system, even the identification of stationary operating points can be a difficult task. However, in an automotive context, the particular systems are developed from generation to generation and so most former versions of the system are available in production vehicles. These systems can be used to get prior information about the system. The identification of the stationary input space can be done by using appropriate measurement periphery like test benches, where the violations of the boundaries are detected and the test bench keeps the system protected from being destroyed.
This section presents two approaches for practical and dynamical DoE measurement. On the one hand, an *offline* method is shown where the stationary input space is used to scale the excitation signals into it (section 2.5.1). On the other hand, an *online* method is shown where a heuristic method allows to get also measurements outside of the stationary space (section 2.5.2).

2.5.1 Dynamical Offline DoE

The idea of the dynamical offline DoE is to obtain a feasible input region and constrain an input signal into it. The feasible input region can be achieved by a stationary system measurement. These stationary samples can be used to compute a hull in which a dynamical excitation is allowed. The original dynamical DoE has the form of a hypercube, thus the trajectory has to be scaled into the drivable hull. The scaling method can be found in Nguyen-Tuong, Bischoff, Imhof & Kloppenburg (2014). The principle of this method is shown in Fig. 2.17.

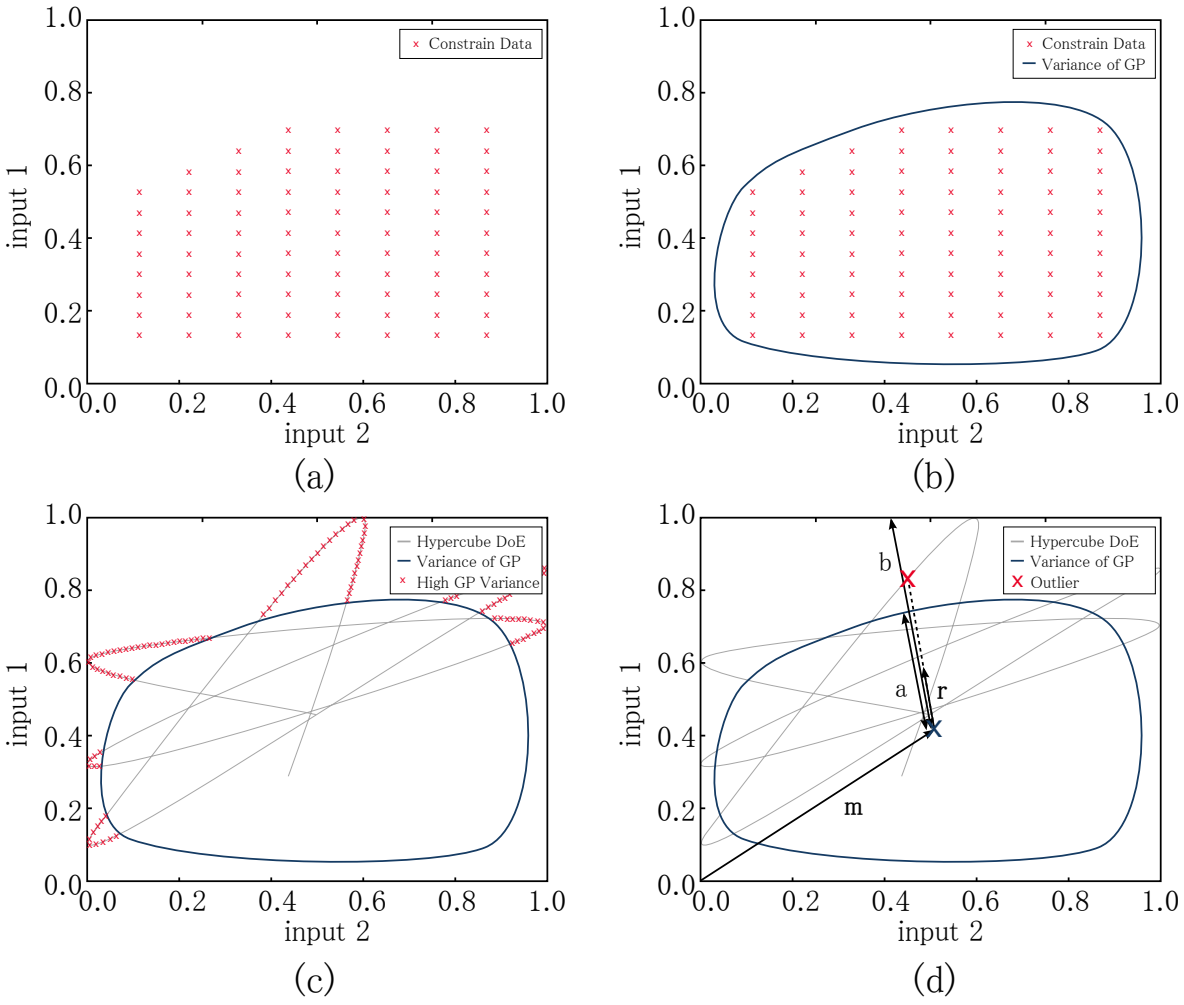


Figure 2.17: Dynamical offline DoE principle: (a) constrain data (b) training result of a GP model (c) classification of samples outside the hull (d) scaling each outlier

In (a) the stationary measured samples are shown. In order to scale a signal into the feasible region, a classification is needed to decide if a query point is inside or outside. A possible solution is shown in (b) using a classification algorithm, for instance the *Gaussian Process Classification* (GPC) (see chapter 3). GPC delivers a multivariate Gaussian distribution of the drivable points and identifies points which are outside the hull by a high GP variance (see Rasmussen & Williams (2006)). In (c) the result of the classification for an arbitrary hypercube DoE is shown. The outliers are marked by red crosses. In the last step the outliers are scaled to the feasible input space (see (d)). The scaling can be done by computing the distance ($|\boldsymbol{a}|$) between the mean ($\boldsymbol{m}$) of the hull with the intersection of the direction vector $\boldsymbol{r}$ with the hull and the distance ($|\boldsymbol{b}|$) between the mean of the hull and the hypercube. Each outlier can thus

be scaled by $\boldsymbol{s} = \boldsymbol{m} + (|\boldsymbol{a}|/|\boldsymbol{b}|)\boldsymbol{r}$.
A result of the method is shown in Fig. 2.18.

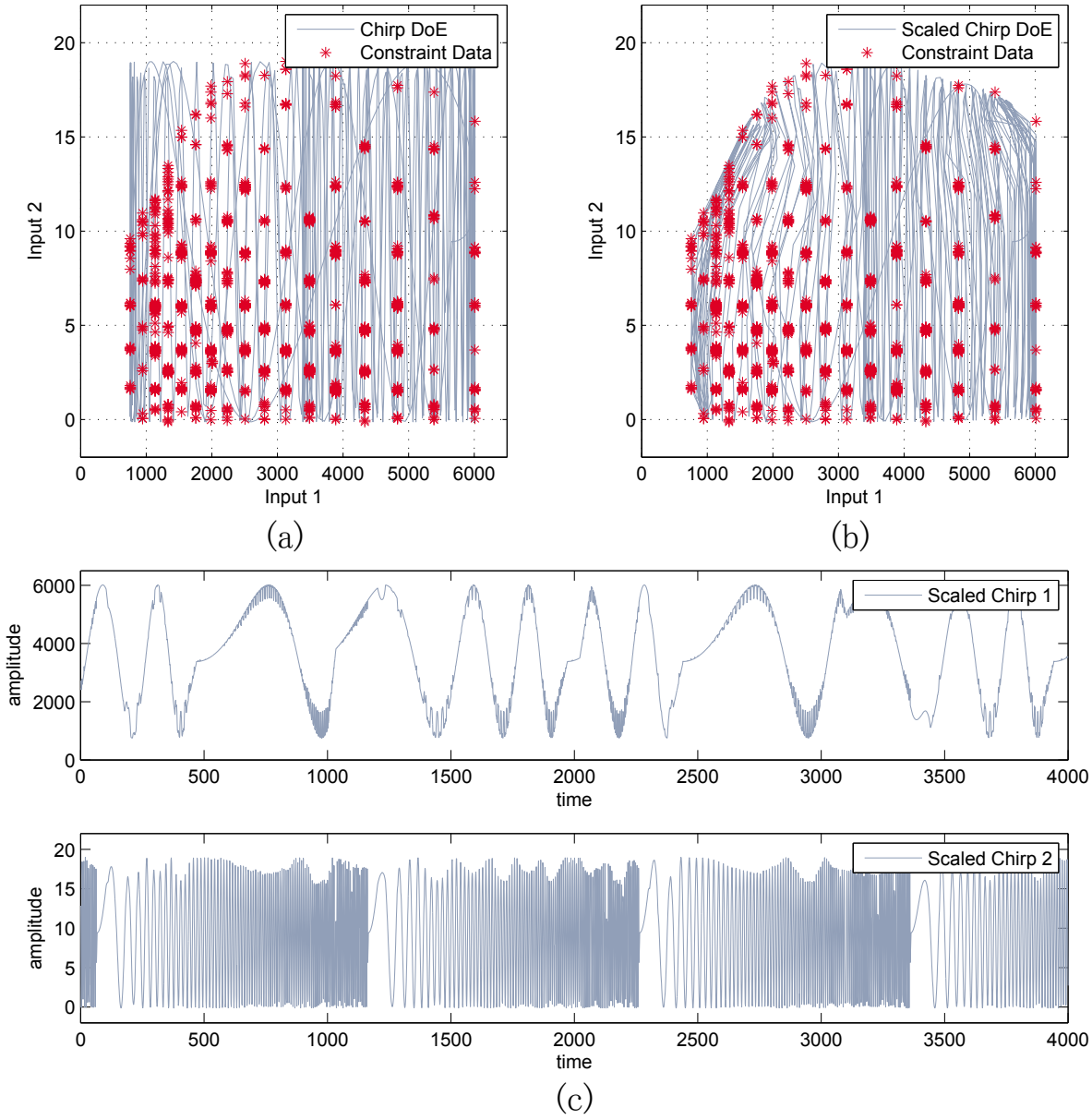


Figure 2.18: Dynamical offline DoE: (a) original chirp DoE and stationary measurement. (b) scaled chirp DoE. (c) time-domain chirp trajectories

In (a) the original chirp DoE plan is shown (blue lines) and a stationary measurement is depicted as red crosses, indicating feasible stationary input space. By computing lines between each trajectory point and the centre of the hull, the intersection of the line and the convex hull and the intersection of the line and the hypercube can all be

determined. The ratio of the line distances defines the scaling factor of the trajectory point. Scaling the original chirp DoE to this hull results in sub plot (b). The adapted DoE plan is shown in (c). In (b) the original hypercube DoE plan is scaled to the feasible input space. Plotting these modified trajectories results in (c). However, given the scaled Chirp signals as transient trajectory, the system can be excited in the region of feasible inputs which minimises the risk of system destruction.
This method of scaling a dynamical DoE plan into a feasible input space guarantees good input space coverage inside the stationary boundaries. In this way a valid model can be achieved. However, Fig. 2.2 denotes that the whole dynamical input space is larger than the dynamical input space inside the stationary boundaries. Therefore this method will not cover the complete dynamic input space.

2.5.2 Dynamical Online DoE

Using offline DoE, as shown in the preceding section, leads to good measurement data inside the convex hull of the stationary boundaries. The drawback of this method is that the globally dynamical input space is usually much larger than the assumed local dynamical input space inside the stationary boundaries (see Fig. 2.2). In order to get samples outside the stationary bounded hull, a concept of an online approach is introduced. The principle of this online DoE approach will be explained using a nonlinear dynamical system with a single input and single output (SISO) structure given as:

$$\begin{aligned} y(k) = &- 0.06[x(k-1) - 0.2y^2(k-1)] \\ &+ 0.1[x(k-2) - 0.19y^2(k-2)] \\ &+ 1.66y(k-1) - 0.701y(k-2) + 0.22. \end{aligned} \tag{2.1}$$

Similar to the offline DoE approach, the first step is to define the stationary drivable input space. Furthermore, the stationary non-drivable maximum boundaries should be defined. This step is identical to the definition of the hypercube boundaries of the offline DoE approach. The idea of the online DoE is to now use a Sobol design, which was initially used for a stationary input space. The Sobol plan can be designed for the dynamical input space by the $u(t)$ and $u(t - T_0)$ space (here T_0 is the sampling time). In Fig. 2.19, a dynamical space filling design is shown. The red line illustrates the stationary input values where the past $u(t - T_0)$ is equal to the current value $u(t)$.

The thin circles are dynamic sample points, whose past values differ from the current values. Each dynamical sample point can be interpreted as a slope, which can be computed as $s = (u(t) - u(t - T_0))/T_0$.

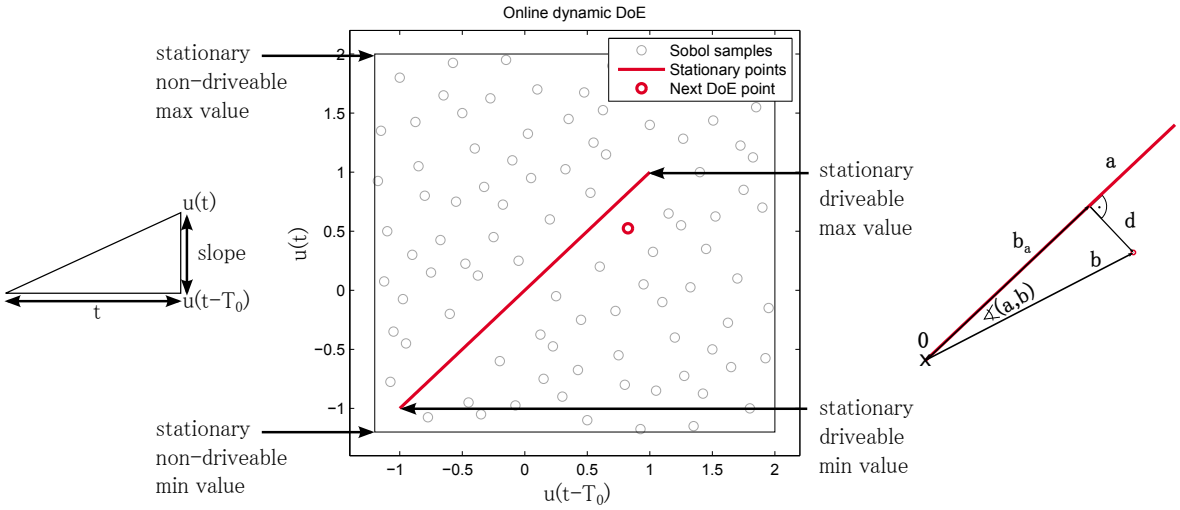


Figure 2.19: Dynamical Sobol plan inside dynamical boundaries

The aim of the dynamic online DoE algorithm is to cover the dynamical input space without violating the system output boundary. It is assumed that the most uncritical dynamic point is the one which is close to the stationary sample points (red line). The closest point can be computed by the smallest vertical distance between the sample points and the stationary line. Fig. 2.19 shows the closest dynamical point for the first iteration (thick circle). On the right hand side of Fig. 2.19 the computation of the distance is depicted. The distance $|\boldsymbol{d}|$ is given by the vectors $\boldsymbol{b}$ and $\boldsymbol{b_a}$ as: $|\boldsymbol{d}| = |(\boldsymbol{b} - \boldsymbol{b_a})|$. $\boldsymbol{b}$ is given by the Sobol samples, the vector $\boldsymbol{b_a}$ is unknown. $\boldsymbol{b_a}$ can be computed by projection as:

$$\begin{aligned} \boldsymbol{b_a} &= \frac{|\boldsymbol{b}|}{|\boldsymbol{a}|} \cos \sphericalangle(\boldsymbol{a}, \boldsymbol{b}) \boldsymbol{a} \\ &= \frac{\boldsymbol{a}\boldsymbol{b}}{|\boldsymbol{a}|^2} \boldsymbol{a} \end{aligned}$$

The aim is to achieve this dynamical point with a measurement. The dynamical point can be interpreted as a slope s, defined by the triangle of $u(t)$ and $u(t - T_0)$ (see Fig. 2.19 left side). In order to receive sample data incorporating the defined slope, a sine wave can be used for excitation. By transforming the particular slope to a

frequency of the sine it can be guaranteed that no greater slopes appear while exciting with the sine wave. Given a sine signal as:

$$Asin(2\pi ft)$$

with f as the frequency, A as amplitude and t as the time, the maximal slopes of the sine appear by the intersection with the x-axis at $0, \pi, 2\pi, ...$ The maximal slope can be computed by the derivative at these intersections:

$$\frac{d}{dt}(A\sin(2\pi ft)) = A2\pi f\cos(2\pi ft)$$

Thus the maximal slope s is given as:

$$s = A2\pi f$$

which gives a frequency of:

$$f = \frac{s}{A2\pi}$$

The idea is to slowly increase the amplitude of the sine signal, while keeping the frequency constant and monitoring the system output. The monitoring is necessary in order to prevent the system from destruction. The excitation of the nonlinear dynamical system (see Eq. 2.1) for the first iteration is shown in Fig. 2.20. The blue ellipsoid in (a) shows the increasing of the sine amplitude in order to reach the dynamic DoE point (red circle). The last sine excitation can be seen in plot (b). It is shown that the sine wave excites the system outside the stationary drivable boundaries. By monitoring the output value, a safe measurement is ensured.

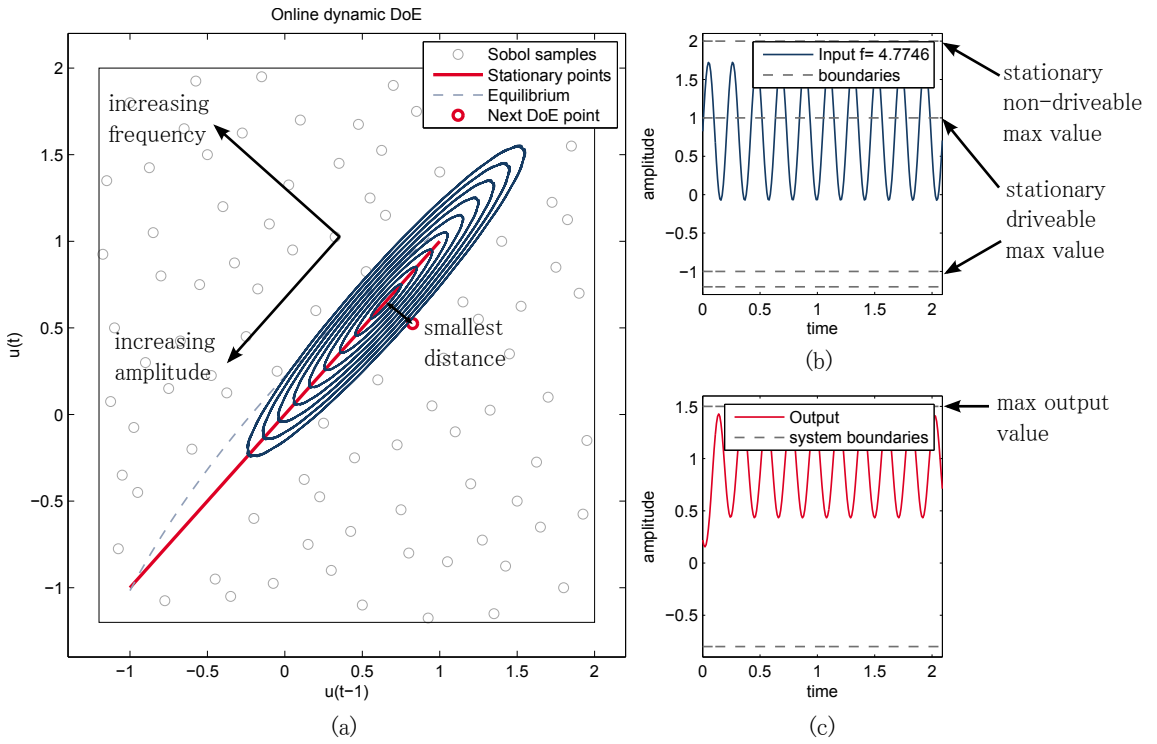


Figure 2.20: First iteration of the dynamical online DoE

The principle of the dynamic online DoE is given in Fig. 2.21.

- S1 and S2: Stationary and maximum boundaries have to be set. The boundaries can be visualised in the dynamic space where the stationary points define a line.
- S3: In order to get a space filling information of the dynamical space a Sobol design can be used to set some targets where measurements are needed.
- S4: A near Sobol point to the stationary line means a small slope. In order to prevent the system from destruction the excitation starts with small slopes and thus the nearest point is computed.
- S5: The slope of the particular Sobol point is used to define a sine wave. The excitation starts with a small amplitude.
- S6: The amplitude of the sine increases until the Sobol point is reached and information have been captured. After the Sobol point is reached, the loop goes back to S4 and the next closest point can be computed.

- S7: The increasing of the sine amplitude is critical, thus the output of the system is monitored. If the increasing of the amplitude results in a too high system output the loop breaks and continues again at S4.

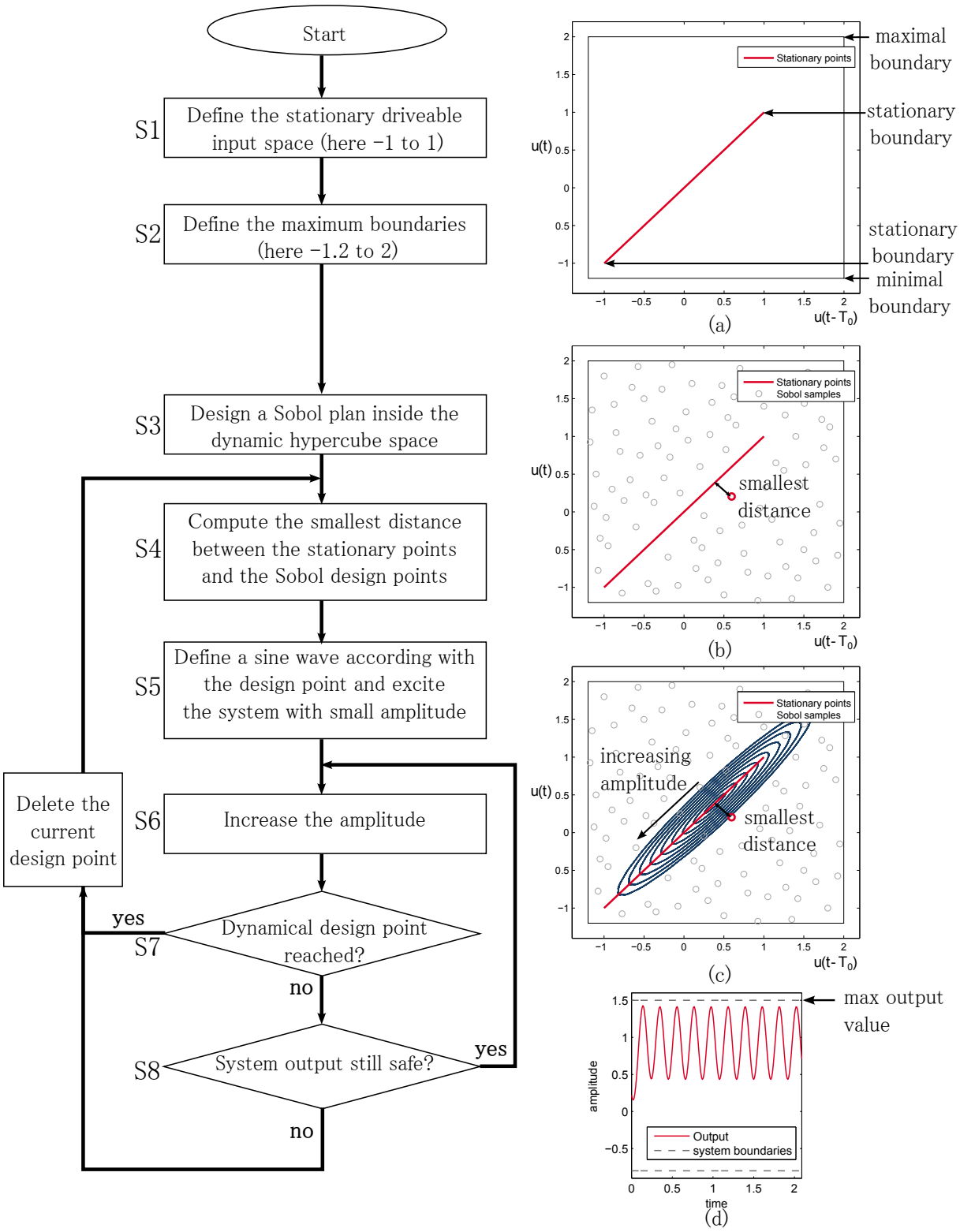


Figure 2.21: Result of the online DoE

The result of the algorithm for the dynamical system (2.1) is given in Fig. 2.22.

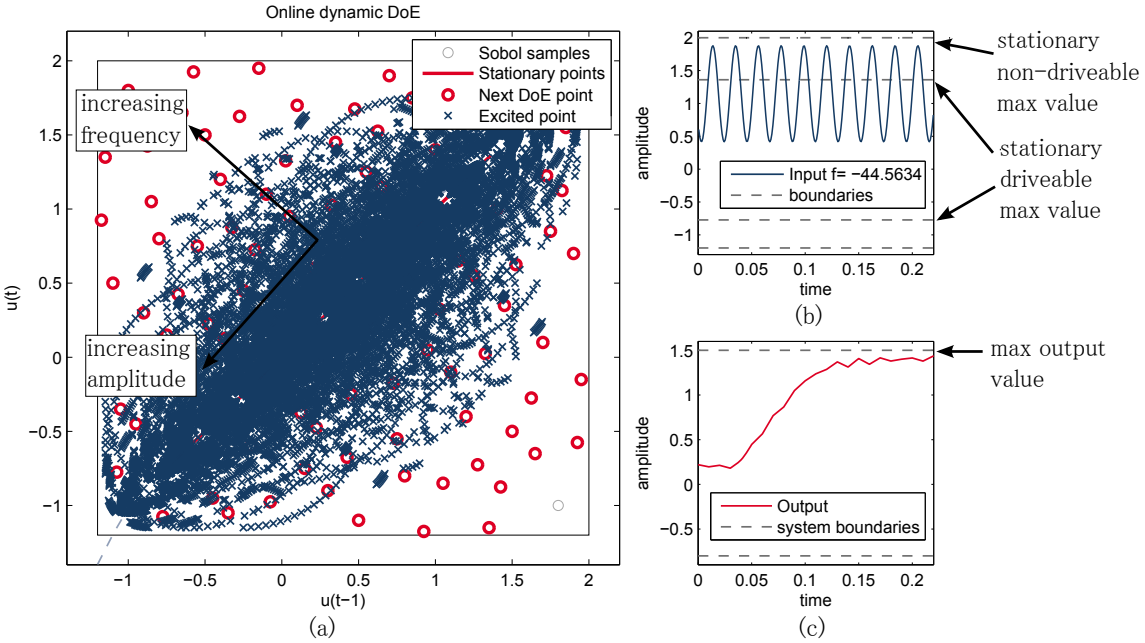


Figure 2.22: Result of the online DoE

It can be seen that sample points are achieved outside the stationary boundaries. Fig. 2.23 shows a comparison of the online DoE result with the offline DoE result. It is important to note that it is not clear in the case of the offline DoE whether all sample points are uncritical regarding the system output.

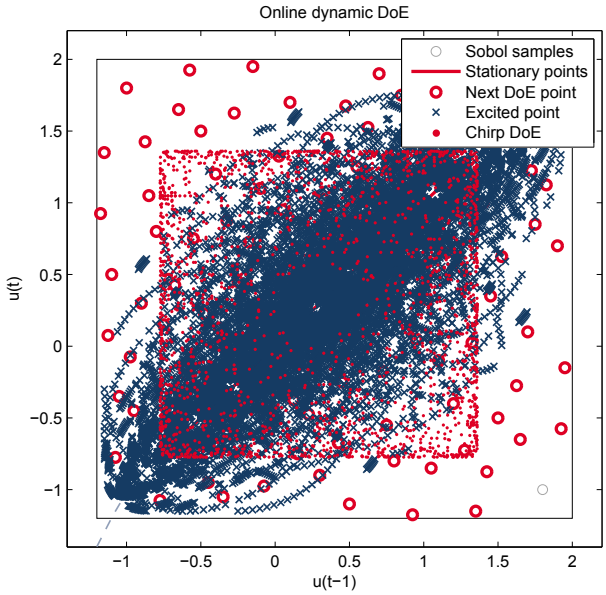


Figure 2.23: Comparison of dynamical input space coverage, offline DoE with Chirps and Online DoE

2.6 Heuristic Space Filling Metric

In the last two sections, different excitation signals were presented and a method for constraining those signals to a feasible input space was introduced. The process of constraining impacts the excitation signal and modifies its spectrum and thus the coverage of the dynamical input space. Therefore, the fulfilment of the space filling property is no longer given. Even if a well-suited excitation trajectory was defined, the real conditions would lead to disturbances and thus to a difference between the artificial signal and the real measurement trajectory. Disturbances of real conditions are, for instance:

- excitation speed over the ECU is limited to a maximal frequency
- communication delays between excitation tool and the ECU
- non-deterministic actuation of the ECU
- excitation over the ECU is not directly over the actuator, but anywhere in the software calibration interface

- synchronisation of different excitation sources are critical
- mean tolerance sensors and small time delays are required

However, it becomes clear that the measurement differs from the initially generated excitation signal and no metric exists to evaluate the measurement for its applicability for the modelling process.
In the following, a possibility to receive this required metric will be introduced. This approach is called *Heuristic Space Filling Metric* (HSFM).

2.6.1 HSFM motivation

To get an intuitive understanding about the motivation of HSFM, a two-dimensional dynamic nonlinear model is given by an extension of Eq. 2.1 in 2D. Thus a *Multiple Input Singe Output* (MISO) system is given. From the identification point of view, the system is a black-box and the aim is to measure the system in order to obtain the nonlinearities and dynamics captured in the data. Therefore, a *shifted chirp* DoE plan is generated. As known from section 2.4.4, the crucial part of the chirp signal is the duration of the excitation in order to obtain a good input space coverage. The first measurement of the system is given in Fig. 2.24. Here, the system was excited by three chirp shifts, each with a duration of ten seconds.

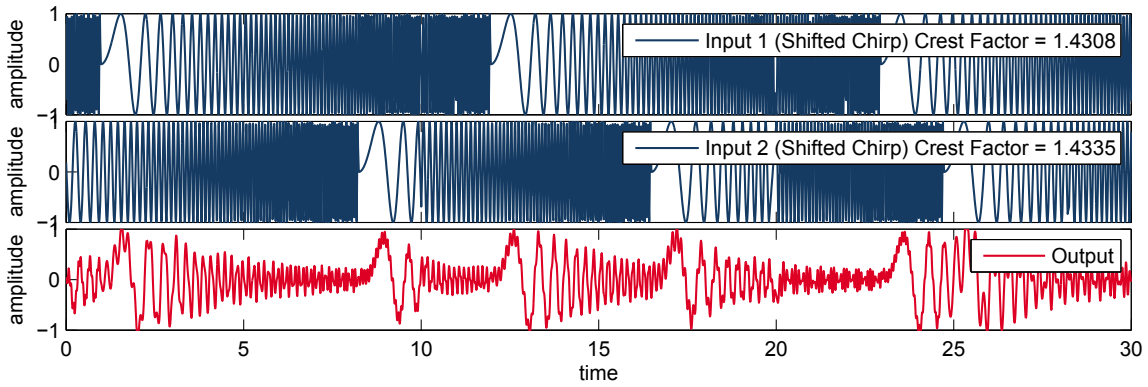


Figure 2.24: Shifted chirp of three shifts, each with a duration of ten seconds

Since it wasn't sure that this measurement was sufficient for the modelling process, a second measurement was taken. This time the chirp was repeated four times, see Fig. 2.25.

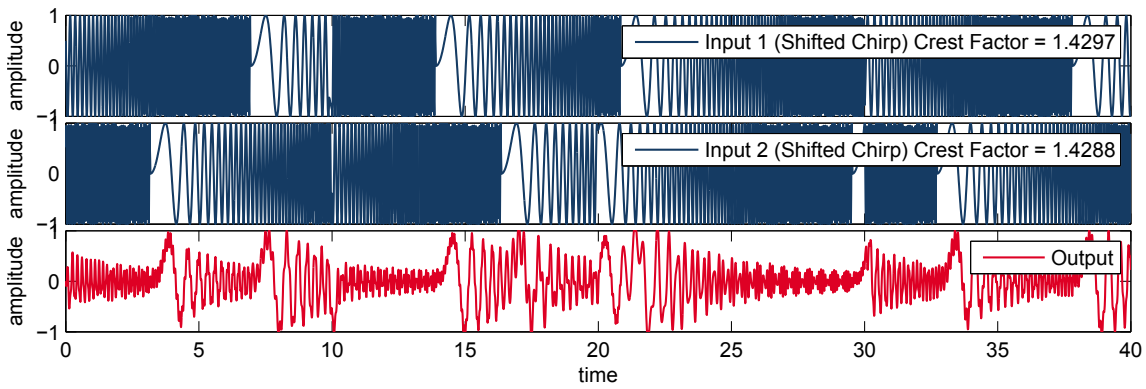


Figure 2.25: Shifted chirp of four shifts with each a duration of ten seconds

Obviously, the question arises: Is the second measurement better than the first one, and if yes, how often must the chirps be shifted to reach the optimum? Additionally: Is a *Sobol-ramp*, an *APRBS-ramp* or a *multisine* DoE perhaps better for this problem? The HSFM will give answers to these questions.

2.6.2 HSFM concept

To reach the requirement of space filling, it is important to aim for an equidistant distribution of the sample points. The maximum number of sample points for a hypercube space can be defined by the number of dimensions d (inputs + output) and the number of necessary samples per dimension s (the optimal number of samples per dimension is dependant on the system complexity). Thus the number of maximal sample points that can be measured, if no boundaries exist in the hypercube, are given by $p = s^d$. Obviously, the maximum value cannot be reached for real applications, since the real system space is no hypercube, but it will be seen that the HSFM is a relative metric and thus the real maximal number of sample points is not necessary to give an answer about the quality of the measurement.
However, given an arbitrary measurement, the distribution of the measurement samples can be analysed relative to the optimal sample distribution and by taking the maximum number of possible Sobol samples into account. The previously defined number of necessary samples per dimension can be used to define the minimal allowed distance (l) between two measured sample points as $l = 1/s$. The measured data can be reduced distance-based, but it is important to mention that an arbitrary erasing of sample points would lead to gaps in the space. Thus the HSFM algorithm only deletes

as many sample points as necessary.
The principle of the erasing process is shown in Fig. 2.26. In the first step (S1) the standard training data are given and are visualised in (a). By defining a maximum density by the user a maximum number of theoretical measurement points are given, see (b). As can be seen in (b), some of the measured data points are close together and thus violate the defined maximum density in step (S2). Now in step (S3) the algorithm loop starts and computes the distances of all training sample points to all the others and counts the number of distance violations. In the first iteration the centre point violates the distance four times, see (c). In the next step (S4) the point with the most violations is deleted, here the centre point. In step (S5) the algorithm checks if further distance violations exist. In this example three more points have small distances between each other and thus the algorithm loop continues at step (S3). Again the sample point with the most distance violations is deleted, in this case two, see (d). If the loop breaks all sample training points fulfill the maximum allowed density, see (e). As can be seen in step (S6), in this example 10 points remain. These 10 points can be used to give a relative impression about the quality of the training data. By computing the ratio of the remaining sample points and the theoretical sample points which were computed in step (S2) a HSFM value is given as $HSFM = s_{left}/s * 100$ and here $23,8\%$
The Matlab code of the HSFM algorithm is given in appendix A.3.1.

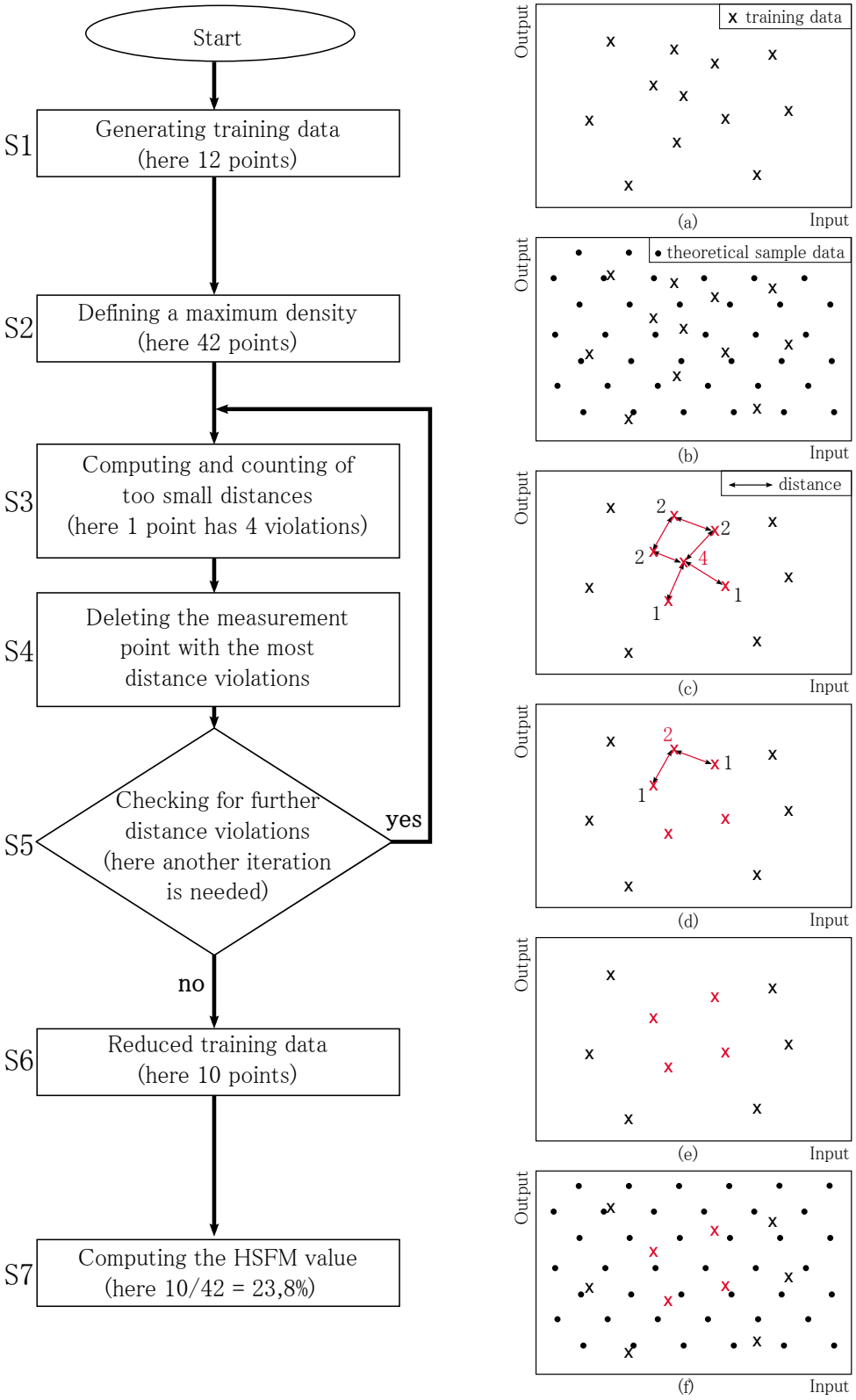


Figure 2.26: HSFM erasing: (a) given a set of training data. (b) defining a maximum point density. (c) first iteration: five sample points violate the minimum distance. (d) second iteration: three sample points violate the minimum distance. (e) remaining sample points. (f) HSFM value as ratio between remaining sample points and maximum theoretical sample point.

Fig. 2.27 gives a 3D overview of the HSFM erasing result. In (a) the grey circles show a Sobol space filling distribution. The blue circles express an arbitrary measurement of the system. In (b) the red circles are the left sample points of the HSFM algorithm. The ratio of grey circles to red circles provides the HSFM value.

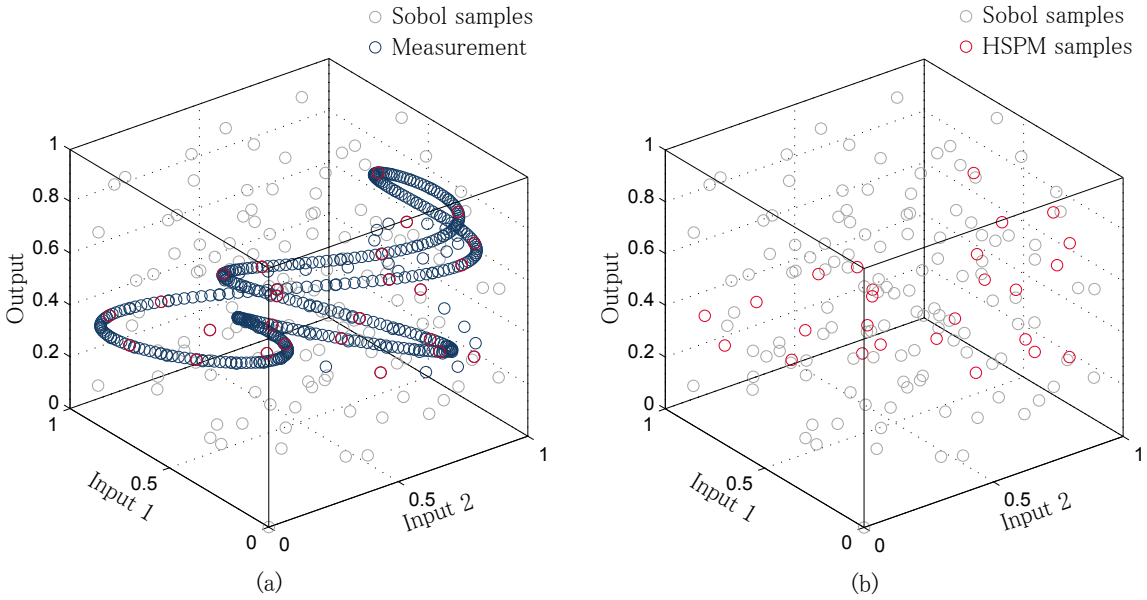


Figure 2.27: HSFM erasing in 3D

Obviously, the HSFM value can be used to analyse any system measurements, as shown in Fig. 2.27 for a 3D example. Note that the HSFM value is especially interesting for higher dimensions, where the distribution of the samples cannot be visually analysed. Given the external dynamic structure of Fig. 2.4, the time delay of the inputs and outputs leads to high dimensional problems, which can be analysed using HSFM. It is thus possible to compute a static value $HSFM_{static}$ where the dimension of the hypercube is $d = inputs + output$ and a dynamic hypercube value $HSFM_{dynamic}$ where the dimension of the hypercube is given by the dynamical structure and thus $d = inputs + time\ delayed\ inputs + time\ delayed\ output + output$. An overview of the different excitation signals and their HSFM is shown in Fig. 2.28. It can be seen in (a) that the APRBS signal has a small HSFM value, especially for the dynamic HSFM value. This means that the coverage of the dynamic input space is worse compared to the other excitation signals. The Ramp excitation in (b) shows a good dynamical coverage, which is expressed by the dynamic HSFM value. However, since the distribution of the ramps is concentrated at the centre of the input space, the static HSFM value is worse compared to the other excitation signals. In this example,

the Multisine signal shows the best HSFM values (c). However, the HSFM strongly depends on the signal parameter, e.g. frequencies, hold times, shift numbers and duration etc.. Thus the shown values are not general expressions of the signal type quality, but can be used to compare any generated trajectories with each other.

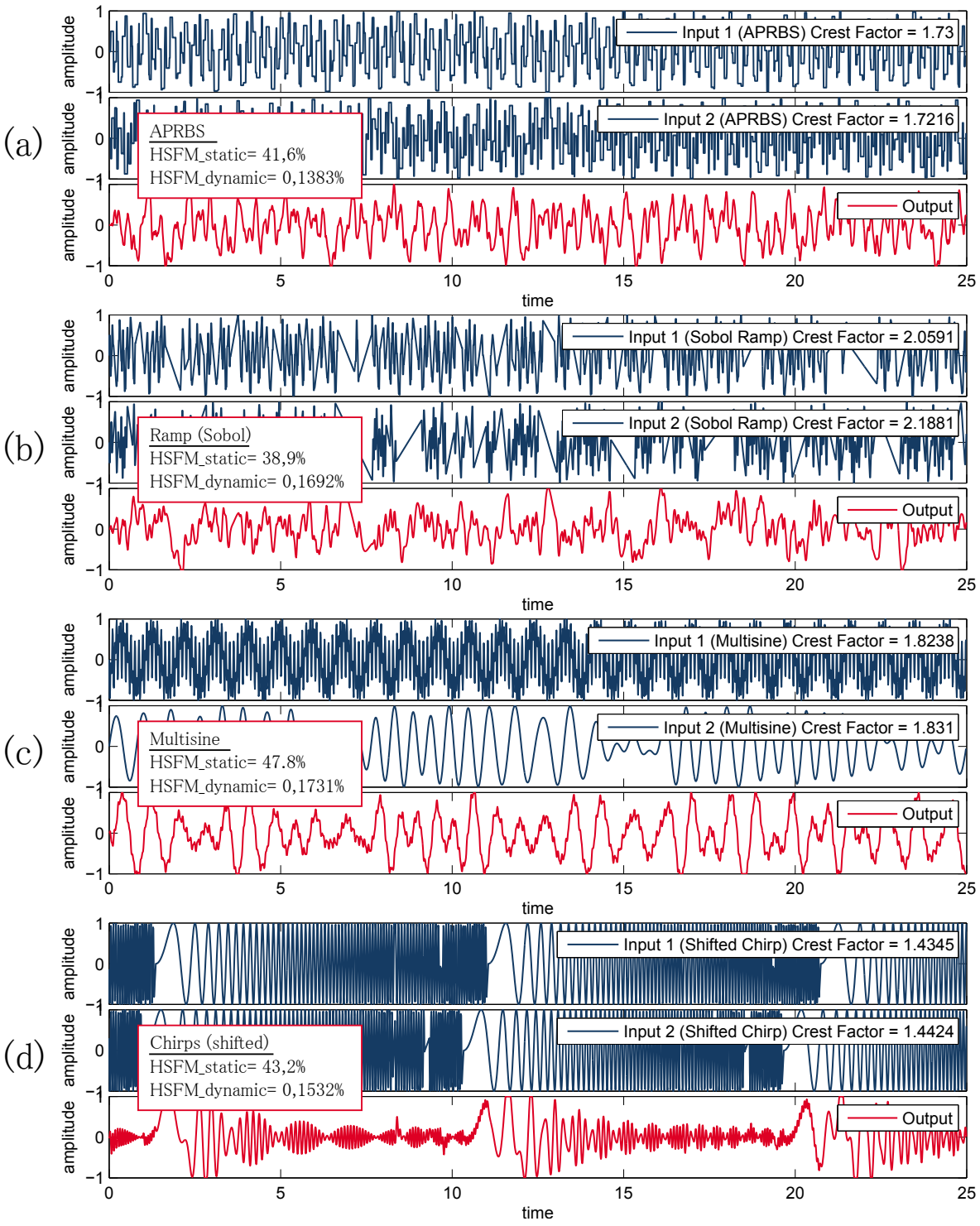


Figure 2.28: HSFM evaluation for different excitation signals: (a) APRBS, (b) Ramp, (c) Multisine and (d) Chirp for a 2D dynamical system

2.7 Summary of Dynamic Design of Experiments

In the beginning of the chapter Dynamic Design of Experiments a theoretical analysis of excitation requirements and signal types was done. It has been shown that multisine and shifted chirp excitations have a high potential to receive appropriate measurement data regarding space filling property.

Signal type	Selectable frequencies	Spectrum	Crest factor	Space Filling distribution
APRBS	--	0	-	-
Ramp	-	-	–	-
Multisine	++	++	++	++
Shifted Chirps	+	++	+	++[1]

[1] after sufficient repetitions

The difficult process is to identify dynamical system boundaries. Thus for offline DoE a method was presented on how to use a stationary input space in the first step of identification and how to scale a dynamical trajectory to this feasible input space. However, it becomes clear that the dynamical space is generally larger than the stationary space and so this method does not lead to globally valid models. A second approach was presented using an online strategy. To achieve dynamical data points, the online method excites with sinus excitation and increases the amplitude while monitoring the output. However, the second method is a theoretical approach and has not been tested on real systems so far.
Overall, it becomes clear that the dynamic DoE step is not straightforward and will not lead to perfect system measurement, covering all dynamical system states. In order to analyse measurement data to determine their suitability for model training, a quality metric was introduced that evaluates the measurement compared to an optimal dynamical Sobol test plan. However, due to the fact that no optimal measurement data will be given, a flexible modelling approach is also needed in order to handle many particular system measurements. The modelling step will be discussed in the next chapter.

3 Data-based Modelling

The efficient usage of virtual test vehicles for model-based ECU calibration requires their early supply in the development process. This requirement can only be fulfilled by a fast delivery of the particular plant models. Though it is advisable, whenever possible, to apply physical models (*white-box models*) it can be a tedious and time-consuming task. Furthermore, not all automotive processes can be explained exactly (e.g. emission formation). A disadvantage of high-precision and also complex physical models, is the high computational effort, and therefore they are not practical for real-time applications.
An attractive alternative to physical models are so-called experimental or data-based models (*black-box models*). Data-based models can incorporate unknown nonlinearities encoded in the sampled data, resulting in more accurate models in practice and in general do not require any a-priori knowledge. Furthermore, the fast simulation speed of the resulting models allows their implementation into real-time simulation environments, such as Hardware-in-the-Loop (HiL) systems, and thus enables a model-based calibration of the related ECU software function (Tietze et al. 2014a). An overview to linear and nonlinear black-box modelling and its usage for practical applications is given in Sjöberg et al. (1995) and Ljung (2007). For the identification of dynamic systems Isermann (2011) gives a good introduction.
In this chapter an introduction to the requirements for modelling nonlinear dynamic systems is given in section 3.1, using polynomial models as an example. In the following sections, most public state-of-the-art modelling techniques will be presented and compared to these requirements. Especially Neural Networks (NN), Local Linear Neuro-Fuzzy Models (LOLIMOT) and Gaussian Process Regression (GPR) will be introduced and analysed. Practical applications and model comparisons are given in von Rango et al. (2012), Hartmann et al. (2013) and Sequenz (2013). It will become clear that GPR is a powerful regression algorithm but that it has some drawbacks for dynamic applications. The GPR algorithm will be explained in detail in section 3.3, including an overview to related work (section 3.3.1), Bayesian framework (section 3.3.2)

and Gaussian Processes (GP)(section 3.3.3.1). The drawbacks of GPR for dynamic problems will be shown in section 3.4 and some advanced approaches will be discussed, dealing with this issue. At the end of this chapter, section 3.4.1 will introduce the Local Gaussian Process Regression algorithm which fulfills the requirements of dynamic modelling. In addition, some artificial applications will be presented.

3.1 Regression for nonlinear dynamics systems

For model-based calibration of the ECU function for a particular system of a vehicle, the behaviour of the system needs to be identified by measuring the inputs and the outputs of that system and in light of the data adapt a model. This method of mapping the input-output relationship is called *Regression*. The earliest form of regression was developed by Gauß in 1795 called *method of least squares*. However, generally speaking, given the measured system inputs and outputs the objective is to minimise an error measure between the system data and the model's behaviour in order to obtain the *best* model. In terms of machine learning this is also called *supervised learning*, see Nelles (2001). Fig. 3.1 shows the configuration of the identification of automotive systems. The measured input data $\boldsymbol{x}$ (here generated by the ECU) and the system output y, which is affected by noise, are used to learn a model by minimising the loss function e between the model output $\hat{y}$ and the true system output y. In order to find the best model, the aim of the modelling process is to represent the true system output, without the measurement noise.

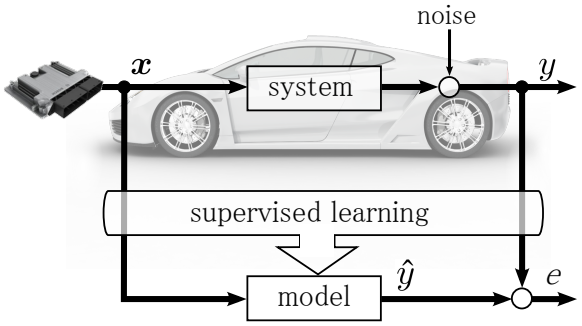


Figure 3.1: Supervised-learning in automotive context

Since the underlying system is dynamical, the identification is focused on generating dynamic models. Thus in the first step the most frequently applied nonlinear dynamic

system modelling and identification approach, which is called *external dynamics* strategy (Nelles 2001), is introduced. Fig. 2.4 of chapter 2 shows its principle architecture. The setup is divided into a static nonlinear approximater $f(\cdot)$, which in this case will be the different model types NN, LOLIMOT or GPR and an *external dynamic filter bank*. Typically, the filters are chosen as simple time delays q^{-1} (Nelles 2001). However, as it can be seen in Fig. 2.4, the external dynamic approach leads to a large number of model inputs. This is a general problem in identification that causes the so-called *curse of dimensionality*. It means that the modelling effort exponentially increases with the number of input dimensions. Generally, a distinction between two different cases of application is made, namely *one-step prediction* and *simulation*. The configurations are shown in Fig. 3.2. Previous system inputs $u(k-i)$ and process outputs $y(k-i)$ are used to predict one or several steps into the future (Nelles 2001). Usually prediction is used for model training, since the y values are taken from the system measurement and are known beforehand. For simulation, the model output $\hat{y}$ itself is used to give information about past system states. The training configuration shown below is also called *Nonlinear Auto-Regression with eXogenous inputs* (NARX), and the simulation configuration is called *Nonlinear Output Error* model (NOE), see e.g. Nelles (2001) or Ljung (2007).

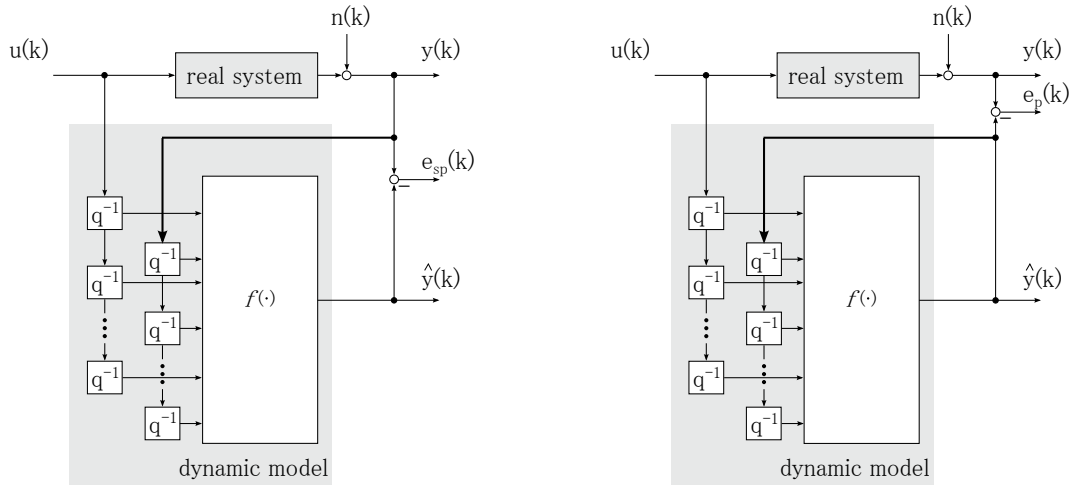


Figure 3.2: one-step prediction as NARX structure (left) and simulation as NOE (right) (Nelles 2001)

The second step now is to choose the appropriate static approximator $f(\cdot)$. In the

following, a polynomial model is used as an example to visualise the requirements for dynamic modelling. The output of a polynomial model is given as:

$$\hat{y} = c_0 + c_1 x + c_2 x^2 + ... + c_m x^m = \sum_{i=0}^{m} c_i x^i \tag{3.1}$$

Given the order of the model, the *regression matrix* $\boldsymbol{X}$ and a parameter vector $\boldsymbol{\theta}$ can be defined as:

$$\boldsymbol{X} = \begin{bmatrix} 1 & x(1) & x^2(1) & \dots & x^m(1) \\ 1 & x(2) & x^2(2) & \dots & x^m(2) \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x(N) & x^2(N) & \dots & x^m(N) \end{bmatrix} \quad \boldsymbol{\theta} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

The parameters can be efficiently estimated by *Least Squares* (Nelles 2001):

$$\boxed{\hat{\boldsymbol{\theta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}}$$

The LS estimate can also be written as:

$$\hat{\boldsymbol{\theta}} = corr\{\boldsymbol{x}, \boldsymbol{x}\}^{-1} corr\{\boldsymbol{x}, \boldsymbol{y}\}$$

and thus be interpreted as the *cross-correlation* of input and output divided by the *auto-correlation* of the input. Given the estimated model parameters, the model output reads:

$$\boxed{\hat{\boldsymbol{y}} = \boldsymbol{X} \hat{\boldsymbol{\theta}}}$$

Fig. 3.3 shows the structure of the polynomial model.

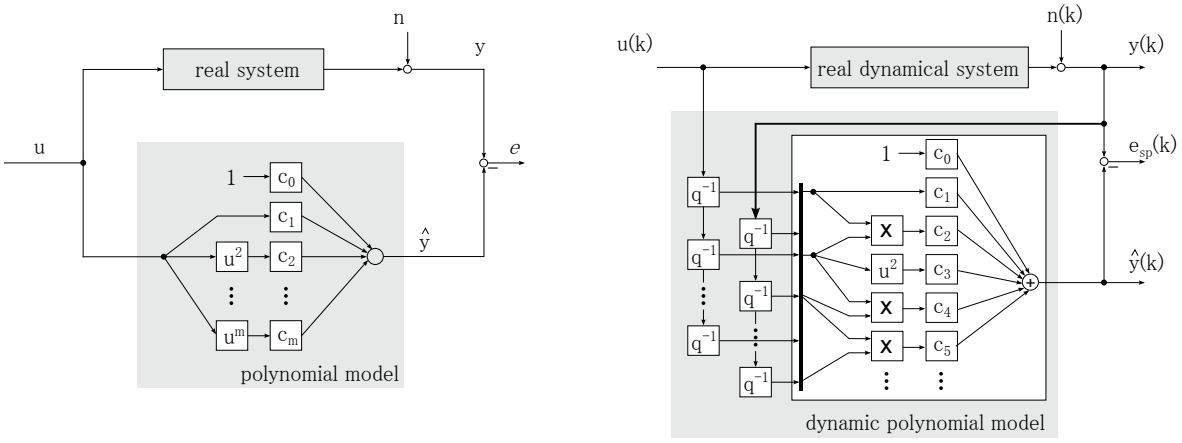


Figure 3.3: Structure of a static polynomial model (left) (Nelles 2001) and a dynamical structure with exemplary regressors (right)

As can be seen from Fig. 3.3 the dynamical structure leads to a strongly increasing number of possible regressors. This general approach of a nonlinear polynomial model with output feedback is called *Kolmogorov-Gabor polynomial*. A special case of the Kolmogorov-Gabor polynomial is the so-called *Volterra-Series Models* where no output feedback is used. The benefit of the Volterra-Series Models is that it is guaranteed to be stable (Nelles 2001). However, if the model structure is defined and the model parameters $\hat{\boldsymbol{\theta}}$ can be estimated. Note that the dynamic NARX structure at least results in a higher input dimension but in general does not change the method of parameter estimation. Thus the delayed inputs will lead to a larger regression matrix and increase the number of parameters. However, one has to consider the question on how to define the order of the polynomials and the order of the dynamical system which leads to the number of delayed inputs. Fig. 3.4 shows the results of a static polynomial model fitting through noisy observations of a function. Three different models are shown, differing in their order m. On the left side, a polynomial model of order 16 is shown. The model is able to match the training points with a small error, but generalises poorly for the test data. This effect of overly complex and flexible models is called *overfitting*. The opposite effect appears on the right hand side. Here, the model complexity is too low and thus not flexible enough to fit the data. This is called *underfitting*. The plot in the middle shows good results, even for training and for test data. The error of training and test data for polynomial models from 1st to 20th order is shown in the lower plot. The training error usually decreases by increasing model flexibility but the model will overfit if the complexity gets too high, in this case at 10th order.

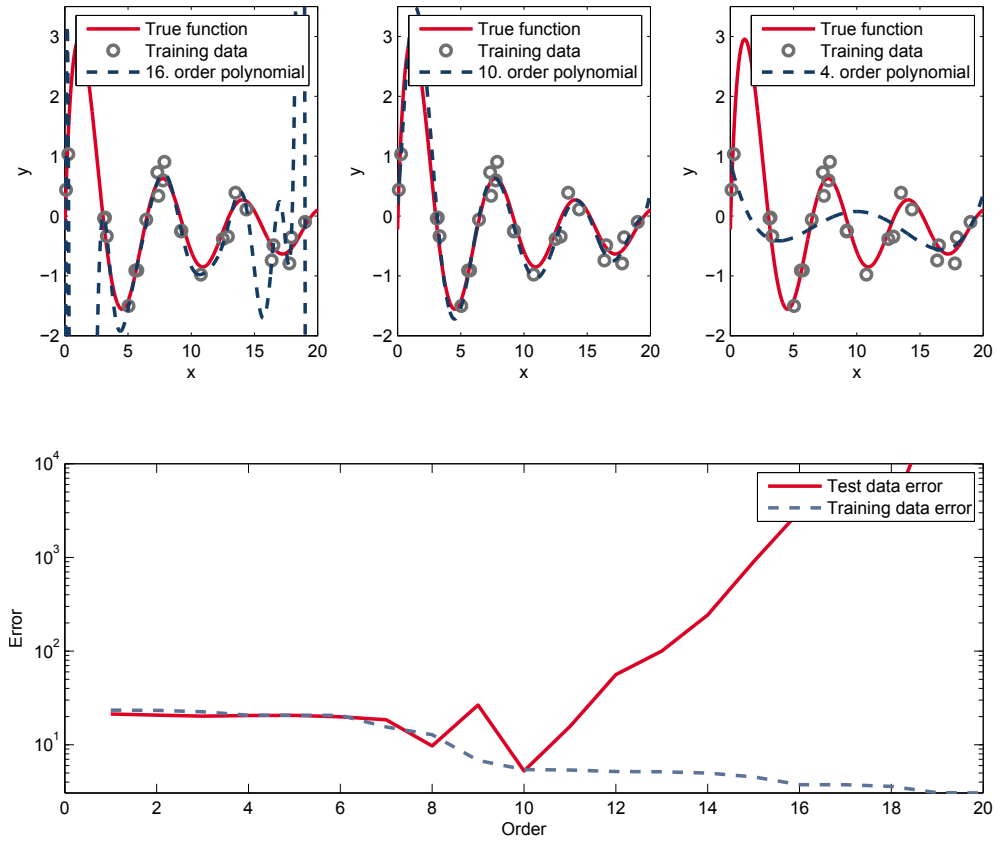


Figure 3.4: 1D example of polynomial fitting

In other words, an excessively complex model leads to a small *bias* but a high *variance*. The difference of the model variance is illustrated well in Fig. 3.4, comparing the 16th and 4th order model. Conversely, low complexity leads to a small *variance* but a high *bias*. A high bias can be seen in the results of the 4th order polynomial. For parametric models this trade-off is commonly known as *bias/variance dilemma*, see Fig. 3.5.

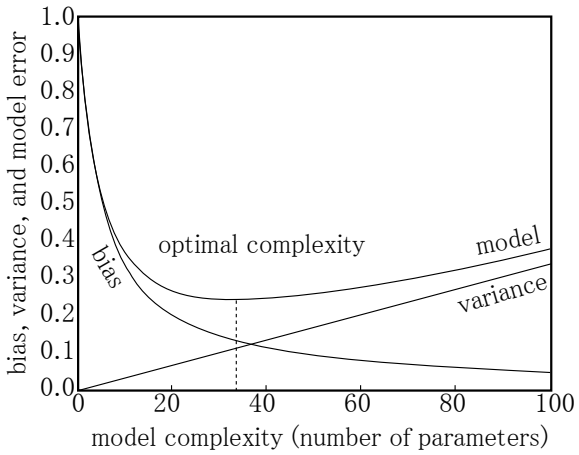


Figure 3.5: Bias/Variance Trade-off (Nelles 2001)

The example of the polynomial model gives an introduction to the requirements of the modelling process. In the following, a detailed comparison of different model types will be given. For this comparison several criteria are defined, due to the external dynamics strategy and are divided into static and dynamic modelling requirements. Furthermore, some requirements regarding the practical usage and the context of automotive systems will be defined. For the static nonlinear approximator, the requirements can be formulated as:

- *Accuracy*: In automotive systems, mechanical effects like system boundaries, leakage or geometry properties often lead to nonlinear system behaviour. Furthermore, if the ECU is part of the underlying system, the ECU function e.g. mode switches leads to effects like hysteresis. However the model must be able to map nonlinear input-output relationships with high accuracy.

- *Parameter estimation*: Data-based models consist of parameters or hyperparameters respectively. These parameters have to be optimised by adequate methods. Therfore a robust optimisation is needed to find the globally optimal parameters.

- *Structure optimisation*: Fitting a model through data results in two tasks, which can be seen from the polynomial model example. The first is the estimation of the parameters, given a fixed model, e.g. a polynomial model of 2nd order. Three parameters have to be estimated for the 1D case. But it is not sure that a polynomial of 2nd order is appropriate or if there exists a model with better properties, e.g. a polynomial of 3rd order. This is called *model comparison*. However, it is scientifically recognised that simpler models should be preferred

unless a more complicated model provides a significant better fit to the data (Loredo 1989). This principle is commonly known as *'Occam's razor'* (Loredo 1989, MacKay 1992*b*). To ensure the *best* model, a modelling approach is needed that finds the optimal complexity, and thus embodies Occam's razor.

- *Sensitivity to noise*: System measurements are usually corrupted by noise. The model needs to consider the correct noise level to avoid overfitting of the data.

- *Interpretation*: A commonly known disadvantage of many data-based models and the reason why they are called black-box models is due to the fact that the parameters are often not interpretable and have no relation to the physics of the underlying system. However a model which delivers interpretable parameters would be beneficial. Furthermore, if the model structure is adapted to the system nonlinearities, it can also improve the interpretability.

- *Incorporation of prior knowledge*: The next commonly known drawback of data-based models is the difficulty of incorporating prior system knowledge or prior beliefs. This is related to the missing physical interpretation of the model. The possibility of incorporating physical knowledge into the model is beneficial and should be pursued. The nomenclature of these models is defined as *grey-box* models.

The external dynamic approach leads to further requirements for the regression algorithm:

- *High dimensional mapping*: Related to the approach of delayed model inputs, the number of inputs strongly increases. Thus the regression algorithm should scale up economically to higher input space dimensions.

- *Large data sets*: Dynamical measurements usually lead to long series of measurements. This is caused by the high dimensionality of the input space and the requirement to cover this space by measuring data, see chapter 2. The model training needs to cope with large data sets.

- *Uneven data distribution*: Measuring a dynamical input space is not as straightforward as described in chapter 2. The measurement data is unevenly distributed and thus the static approximator needs to be robust against unevenly distributed data.

- *Expression of uncertainty*: One of the most important requirements is also caused by the high dimensionality and the challenge of generating input space covering measurement data. However it should become clear from chapter 2 that it is utopian to expect globally optimal system measurements from the DoE step. There will be gaps in the input space and especially the dynamical boundaries will not be captured completely. However, these drawbacks are justifiable, if a metric exists which delivers a reliable model uncertainty during usage.

- *Weakening extrapolation*: From the external dynamics approach in Fig. 2.4 it can be seen that the recursive model output can lead to unstable model behaviour. This is typical when the model extrapolates and the model output amplifies the instability. A strategy that weakens extrapolation behaviour would improve model stability.

Lastly, the requirements for practical usage with respect to model-based ECU calibration are taken into account:

- *Iterative modelling*: The fact that measurement data does not cover the whole dynamical input space implies the fact that the received model will not be globally valid. Thus modelling should be seen as a process and provide an iterative modelling procedure to adapt the model when new measurements arrives.

- *Model recalibration*: Although the aim of the identification is to generate a global model, the usage of the model is afterwards restricted by the ECU calibration. Thus, the model should fulfill high accuracy requirements in the relevant areas. It is thus advisable to measure especially in these areas. However, these areas are often not known beforehand and so a model recalibration is needed when receiving new training data in the desired area.

- *Noise variations*: It is typical for sensors to be accurate in an area where the system is controlled, e.g. oxygen sensor at lambda = 1 for a gasoline engine. The sensor is constructed to have low noise in the area close to 1, because it is necessary for appropriate control. For very lean combustion, the sensor becomes imprecise and noisier. An example is given in chapter 4. However, the regression algorithm should be capable of dealing with noise variations. Also, the system itself can be noisy. When one considers a combustion engine the noise level of the engine, changes depending on load and speed.

- *Adaptation of local complexity*: Local effects appear, e.g. when the engine changes operation modes. For data-based models, local effects mean a change of complexity. In the polynomial system, for example, e.g. the global system can be modelled by a polynomial of 2nd order, but the local effect requires a model of 3rd order. The modelling approach therefore needs a possibility for the adaptation of complexity.

- *Adaptation of local changing dynamics*: Local effects can also be caused by a change of the system's dynamics. From the physical view, if the order of the dynamical system changes, the external dynamic structure of the data-based model has to be adapted to this local effect.

- *Simulation speed*: The target for the model is a HiL system which requires real-time models. Thus, the simulation speed of the data-based model should be fast and require low resources.

- *Requirement of memory*: A further possible target of the data-based models is the ECU itself. Instead of using the accurate model to calibrate the ECU function, a replacement of the ECU function by the data-based model is possible. Therefore the memory requirement should be small.

- *Online adaptation*: Online modelling is an interesting application for efficient modelling due to dynamic identification. Here, a rapid training and the possibility to adapt the model when new data arises is required.

- *Availability of ECU function blocks*: The replacement of the ECU function further requires that the mathematical operations are available in ECU function development.

- *Usability*: In order to achieve a broad usage for calibration tasks, the modelling should be easy and intuitive.

3.2 Model comparison

The requirements for a data-based modelling algorithm for the identification of dynamic nonlinear systems were presented in the preceding section. In this section, the most commonly used regression algorithms, namely Neural Network, Local Linear

Model Tree and Gaussian Process Regression will be analysed with respect to the requirements.

3.2.1 Polynomial Model

Dynamic nonlinear modelling using a polynomial model: The polynomial model has a simple structure and its parameters can be easily and rapidly optimised by the *Least Squares* method. However, the number of parameters grows rapidly with increasing input dimensionality and thus polynomial models are not well-suited for high dimensional problems. Further, the interpretation of the model is very low and so the incorporation of prior knowledge is hardly possible. A main problem of polynomial models is the structure optimisation for high dimensional problems due to the huge number of potential regressors (Nelles 2001). Here, so-called *subset selection* techniques can be applied. For complex problems, high order polynomials tend to overfit and are not practical. Furthermore, the global modelling approach leads to a very inflexible modelling chain. The polynomial model is able to provide a model uncertainty by taking the input data applied for training into account. This approach is called *errorbars*. Generally speaking, a model that was estimated from data can be expected to be good in regions where the data was dense and to be poor in regions where the data was sparse (Nelles 2001). Obviously, the parameter covariance matrix $cov\{\hat{\boldsymbol{\theta}}\}$ determines the accuracy of the model output for a given input (Nelles 2001).

$$\begin{aligned} cov\{\hat{\boldsymbol{y}}\} &= E\left\{\left(\hat{\boldsymbol{y}} - E\{\hat{\boldsymbol{y}}\}\right)\left(\hat{\boldsymbol{y}} - E\{\hat{\boldsymbol{y}}\}\right)^T\right\} \\ &= E\left\{\left(\boldsymbol{X}\left(\hat{\boldsymbol{\theta}} - E\{\hat{\boldsymbol{\theta}}\}\right)\right)\left(\boldsymbol{X}\left(\hat{\boldsymbol{\theta}} - E\{\hat{\boldsymbol{\theta}}\}\right)\right)^T\right\} \\ &= \boldsymbol{X}E\left\{\left(\hat{\boldsymbol{\theta}} - E\{\hat{\boldsymbol{\theta}}\}\right)\left(\hat{\boldsymbol{\theta}} - E\{\hat{\boldsymbol{\theta}}\}\right)^T\right\}\boldsymbol{X}^T \\ &= \boldsymbol{X}cov\{\hat{\boldsymbol{\theta}}\}\boldsymbol{X}^T \end{aligned} \tag{3.2}$$

Here, the covariance matrix of the parameter estimate $cov\{\hat{\boldsymbol{\theta}}\}$ describes the accuracy of the estimated parameters. The errorbars are then given as the model output $\hat{\boldsymbol{y}}$ plus and minus the standard deviation of the estimated output, which are diagonal entries of $cov\{\hat{\boldsymbol{y}}\}$:

$$\hat{\boldsymbol{y}} \pm \sqrt{diag(cov\{\hat{\boldsymbol{y}}\})}$$

The errorbars allow the estimation of the model accuracy for a given input, but it is important to note that these estimations are calculated under the assumption of a

correct model structure and so the errorbar does not indicate if the model structure is correct or not. An example of the errorbar of a polynomial model is shown in Fig. 3.6.

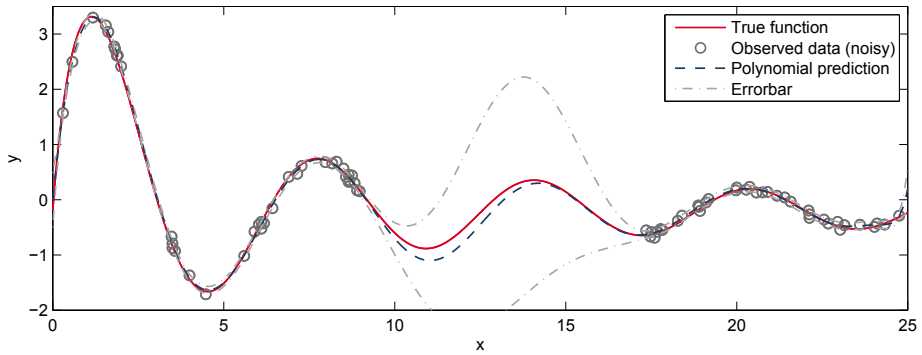


Figure 3.6: Errorbar of a polynomial model

However, polynomial models can easily be implemented as an ECU function, since the mathematical operations are restricted to multiplications and summations and only the parameters have to be stored in the ECU memory. An overview of all criteria is shown in the table of section 3.2.5.

3.2.2 Neural Networks

Artificial neural networks are motivated by the biological structures in the brains of humans and animals (Nelles 2001). The principle of NN is to use simple units in a large number and connect these units to a network. Two classes of neural networks received considerable attention in the area of artificial neural networks namely *multilayer neural networks* and *recurrent neural networks* (Narendra & Parthasarathy 1990). Fig. 3.7 shows the basic structure of these networks. Basically the network consists of an *input layer*, one or many *hidden layers* and one *output layer*. The hidden layer consists of *hidden layer neurons* which typically consist of a weighted sum and a nonlinear function. The output neuron typically is a linear combination of the hidden layer basis functions with an additional offset, which is sometimes called *Bias* (Nelles 2001). The most widely known architecture of a multilayer neural network is called *Multilayer Perceptron* (MLP). Fig. 3.7 shows the static and dynamcial structure. The MLP network is designed of *perceptrons* consisting of a weighted sum of the inputs and the bias combined with a nonlinear so-called *activation function*. The sigmoid function is a common choice for the activation function (Nelles 2001).

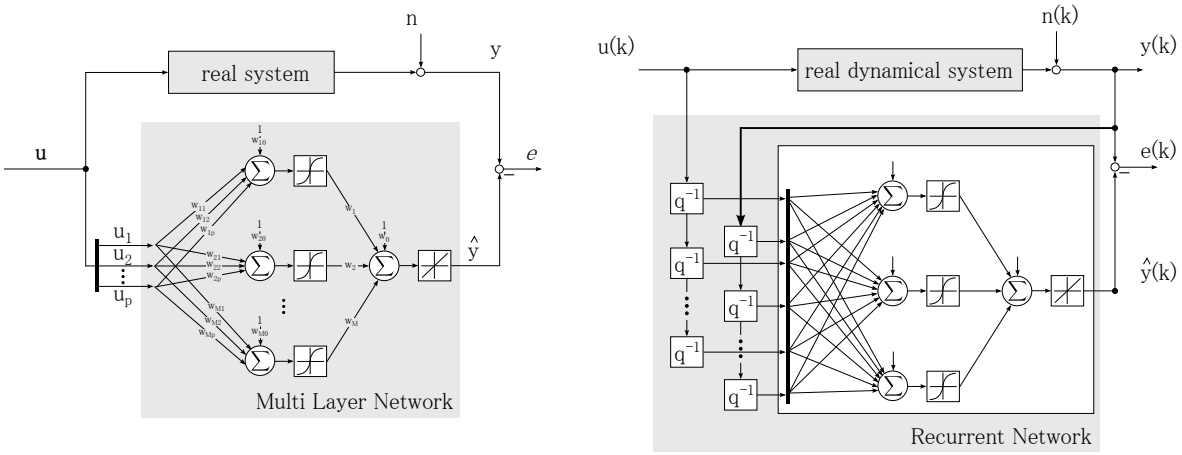


Figure 3.7: Structure of a MLP network

The model output of the MLP with one hidden layer is given as:

$$\hat{y} = \sum_{i=0}^{M} w_i \Phi_i \left(\sum_{j=0}^{p} w_{ij} u_j \right) \quad \text{with} \quad \Phi_0(\cdot) = 1 \quad \text{and} \quad u_0 = 1 \tag{3.3}$$

Training the MLP network means optimising the weights of the output layer and of the hidden layer. The most famous training algorithms are called *Levenberg-Marquardt nonlinear least squares* and *Backpropagation*.

Dynamic nonlinear modelling using the MLP algorithm: Nelles (2001) assesses the MLP networks as well-suited for the external dynamic approach, since MLP networks are able to find the main directions of the nonlinearity of a system and thus can circumvent the *curse of dimensionality*, see section 3.1. MLP networks are relatively insensitive with respect to a too high choice of dynamic orders, because they can cope well with redundant inputs by driving the corresponding hidden layer weights towards zero (Nelles 2001). Furthermore, MLP networks can handle uneven data distribution, which is typical for dynamic systems (see section 2). This is possible because the optimisation of the hidden layer weights transforms the input axes in a suitable coordinate system anyway (Nelles 2001). However, the MLP networks have many disadvantages caused by their complex structure; for instance, many poor local optima exist. The training effort of a MLP network is quite high and a nonlinear optimisation technique has to be used. The parameters are not interpretable and it is generally not possible to incorporate prior system knowledge. Moreover, the results crucially depend on the parameter initialization (Nelles 2006).

3.2.3 LOLIMOT and HILOMOT

LOLIMOT stands for Local Linear Model Tree and is a multi-model approach, which composes local affine models by normalised Gaussian weighting functions to a global model output (Sequenz 2013). LOLIMOT was presented in Nelles et al. (1996) and Nelles (1997) and belongs to the class of incremental tree-construction algorithms since for each iteration a new local linear model (LLM) is added to the model (Nelles 2001). Each local model belongs to a validity region, which is generated by axis-orthogonal splits of the input space. The global model output is calculated by the weighted sum of the local model outputs multiplied by the weighting function (Sequenz 2013), see Fig. 3.8.

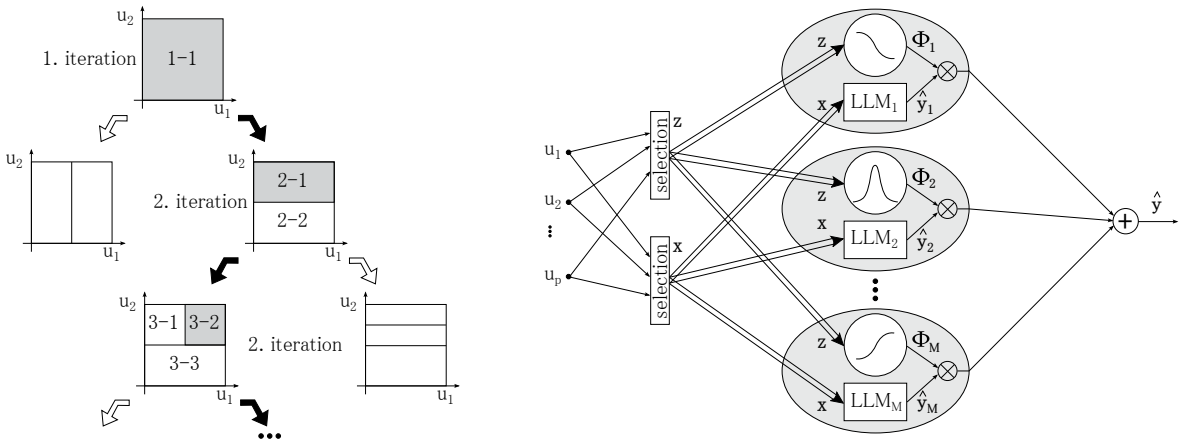


Figure 3.8: Tree construction algorithm and model structure of LOLIMOT (Nelles 2001)

The output of the LLMs is given as:

$$\hat{y}_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \tag{3.4}$$

where w_{ij} denote the LLM parameter for neuron i (Nelles 2001). The global output of the LOLIMOT network is the sum of each LLM weighted by a validity function Φ_i:

$$\hat{y} = \sum_{i=1}^{M} \hat{y}_i(\boldsymbol{x})\Phi_i(\boldsymbol{z}) \tag{3.5}$$

Note that for *identical input spaces* the LLMs and the validity functions depend on the same variables, i.e. $\boldsymbol{x} = \boldsymbol{z} = \boldsymbol{u}$ (Nelles 2001). For a consistent output of the global

network (so-called *partition of unity*, the validity function $\Phi_i(\boldsymbol{z}$ has to be normalised so that:

$$\sum_{i=1}^{M} \Phi_i(\boldsymbol{z}) = 1$$

The network interpolates between different LLMs with the validity functions (Nelles 2001). Typically the validity functions are normalised Gaussians with centre coordinates c_{ij} and standard deviations σ_{ij}, given as:

$$\Phi_i(\boldsymbol{z}) = \frac{\mu_i(\boldsymbol{z})}{\sum_{i=1}^{M} \mu_i(\boldsymbol{z})}$$

with

$$\mu_i(\boldsymbol{z}) = \exp\left(-\frac{1}{2}\frac{(z_1 - c_{i1})^2}{\sigma_{i1}^2}\right) \cdot \ldots \cdot \exp\left(-\frac{1}{2}\frac{(z_p - c_{ip})^2}{\sigma_{ip}^2}\right)$$

The parameters of the local models are given by the weights w_{ij} in Eq. 3.4. Thus a linear optimisation problem arises which can be solved by *least squares* optimisation as shown in section 3.1. Thus, the parameters of each local model can be estimated *simultaneously* by a *global* estimation approach (Nelles 2001):

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

or *separately* by a *local* estimation approach, given as:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}_i^T\boldsymbol{Q}_i\boldsymbol{X}_i)^{-1}\boldsymbol{X}_i^T\boldsymbol{Q}_i\boldsymbol{y}$$

with

$$\boldsymbol{Q}_i = \begin{bmatrix} \Phi_i(\boldsymbol{z}(1)) & 0 & \cdots & 0 \\ 0 & \Phi_i(\boldsymbol{z}(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \Phi_i(\boldsymbol{z}(N)) \end{bmatrix}$$

and regressor matrix $\boldsymbol{X}_i$ and the measured output values $\boldsymbol{y}$ as:

$$\boldsymbol{X}_i = \begin{bmatrix} 1 & x_1(1) & x_2(1) & \cdots & x_p(1) \\ 1 & x_1(2) & x_2(2) & \cdots & x_p(2) \\ \vdots & \vdots & \ddots & \vdots & \ddots \\ 1 & x_1(N) & x_2(N) & \cdots & x_p(N) \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix}$$

The local estimation approach increases the flexibility of the model and thus the bias error, but reduces the variance error (Nelles 2001). For detailed discussion of global

and local parameter estimation, see Nelles (2001). Given the estimated parameters $\hat{\boldsymbol{w}}$ the model given in Eq. 3.5 now reads:

$$\begin{aligned}\hat{y} &= \sum_{i=1}^{M} \hat{y}_i(\boldsymbol{x})\Phi_i(\boldsymbol{z}) \\ &= \sum_{i=1}^{M} \boldsymbol{Q}_i \boldsymbol{X}_i \hat{\boldsymbol{w}}\end{aligned}$$

Given the LOLIMOT architecture as a static approximator model, it can be used for dynamic modelling by pursuing the *external dynamics approach* of section 3.1, by setting

$$\boldsymbol{x} = \boldsymbol{\varphi}(k) \quad \text{and} \quad \boldsymbol{z} = \boldsymbol{\varphi}(k)$$

with

$$\boldsymbol{\varphi}(k) = [u_1(k-1)\ ...\ u_1(k-m)\ ...\ u_p(k-1)\ ...\ u_p(k-m)\ y(k-1)\ ...\ y(k-m)]^T$$

Fig. 3.9 gives an example of a LOLIMOT network with an external dynamic approach.

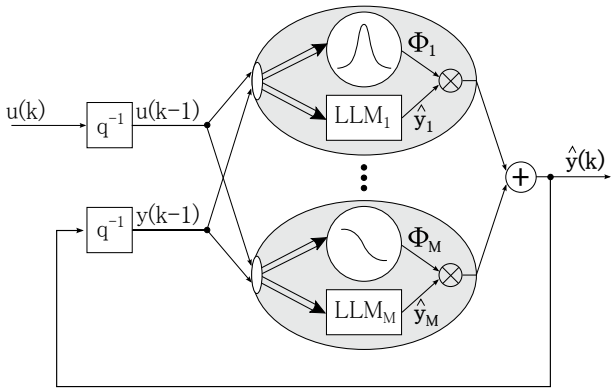


Figure 3.9: LOLIMOT with external dynamics approach (Nelles 2001)

Dynamic nonlinear modelling using a LOLIMOT and HILOMOT algorithm:

Since the local models of the LOLIMOT architecture consist of polynomials, the parameter estimation is quite efficient and thus the training speed is very high. The nonlinear structure parameters result from a heuristic approach which is generally automated and thus provides a high usability. The axes-orthogonal partitions of the input space deliver the interpretability of the underlying system. Furthermore, the inputs of the LOLIMOT model can be declared as linear or nonlinear by the user,

thus an incorporation of prior system knowledge is possible. Further incorporation of prior knowledge can be done by the distinction of the inputs to be part of the model or to be part of the validity function, see Eq. 3.5. Thus, the input space of the linear model can be reduced, which counteracts the curse of dimensionality. However, the local architecture is beneficial for practical applications. Here it is possible to adapt the model efficiently when new data arises. The performance of LOLIMOT degrades more and more with an increasing dimensionality of the premise input space (Hartmann et al. 2013).The axis-othogonal partitioning restricts the flexibility of the model. Thus LOLIMOT leads to relatively large number of local linear models for an adequate modelling (Nelles 2001). A more efficient modelling approach delivers an axis-oblique decomposition, called *Hierarchical Local Model Tree* (abbreviated as HILOMOT). Based on the theory of *hinge functions* presented in Breiman (1993), and the extension to *smooth hinge functions* introduced in Pucar & Millnert (1995), Ernst (1998) presented the approach of *hinging hyperplane trees* for the approximation and identification of nonlinear systems. Ernst (1998) introduced so-called *generalized hinging hyperplanes* where the input space partitioning is independent of the local models. Thus, only the parameters of the two local models that are newly created by a split should be reestimated (Nelles 2006). The construction algorithm of HILOMOT is shown in Fig. 3.10. Compared to LOLIMOT, HILOMOT is beneficial regarding high dimensional mappings which result in more accurate models.

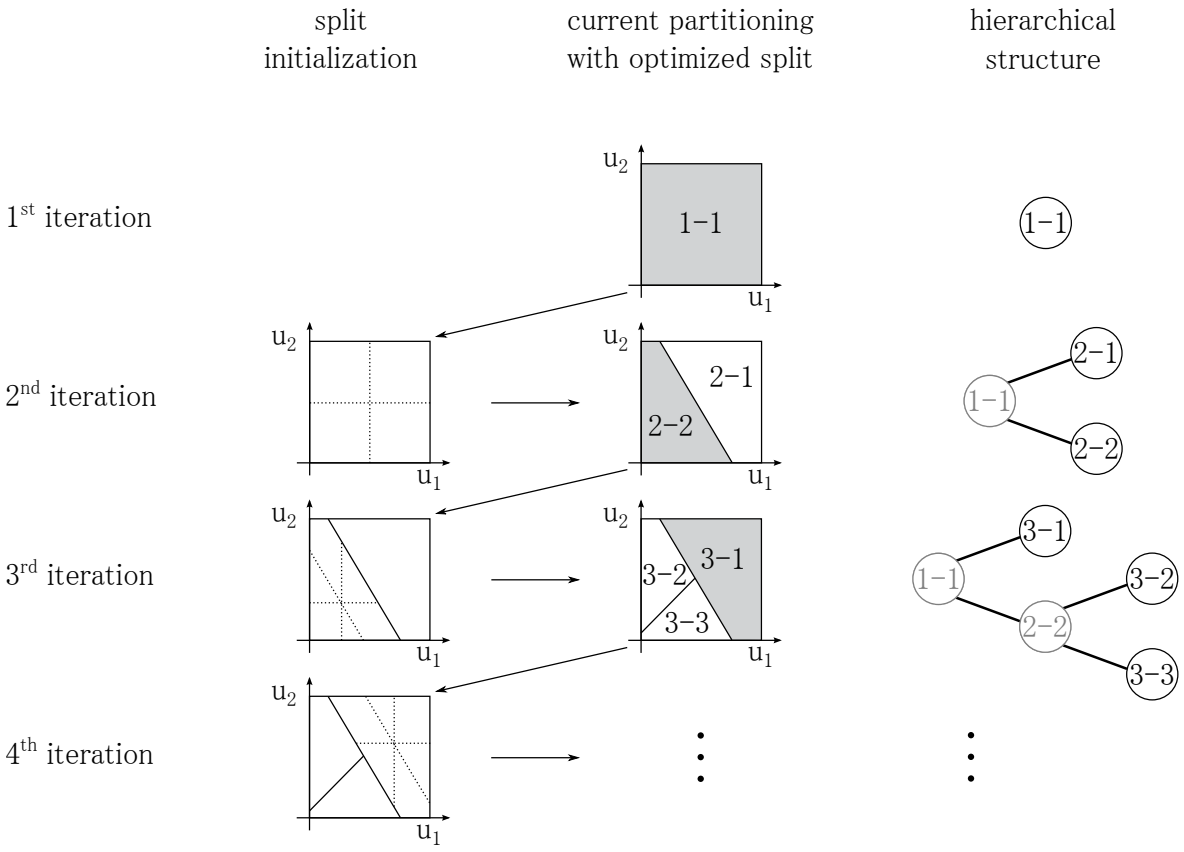


Figure 3.10: Four iterations of the HILOMOT construction algorithm (Hartmann et al. 2013)

3.2.4 Gaussian Process Regression

The GPR theory will be discussed in detail in chapter 3.3. This section presents a brief model overview. GPR is a powerful Bayesian and non-parametric modelling approach which assumes Gaussian distributions on function classes and sampled data. The probabilistic approach allows information to be given about the predictive uncertainty of the model with respect to prior beliefs of the function properties. Nonparametric means that the GPR model contains no parameters to fit the data through the measurement values as opposed to the weights c_i of the polynomial model, see Eq. 3.1. The only parameters that the GPR model contain are so-called *hyperparameters*, which control the complexity of the model. For instance, the order of the polynomial model can be seen as a hyperparameter since the polynomial order controls the flexibility of the model. However, the GPR model outputs are generally samples from a *Gaussian Process*, defined by a *mean function* and a *covariance function*, see Fig. 3.11. Instead of minimising an error to optimise the model parameters as shown for the polynomial

model, MLP and LOLIMOT, the probabilistic approach maximises the probability of the model values, given the measurement.

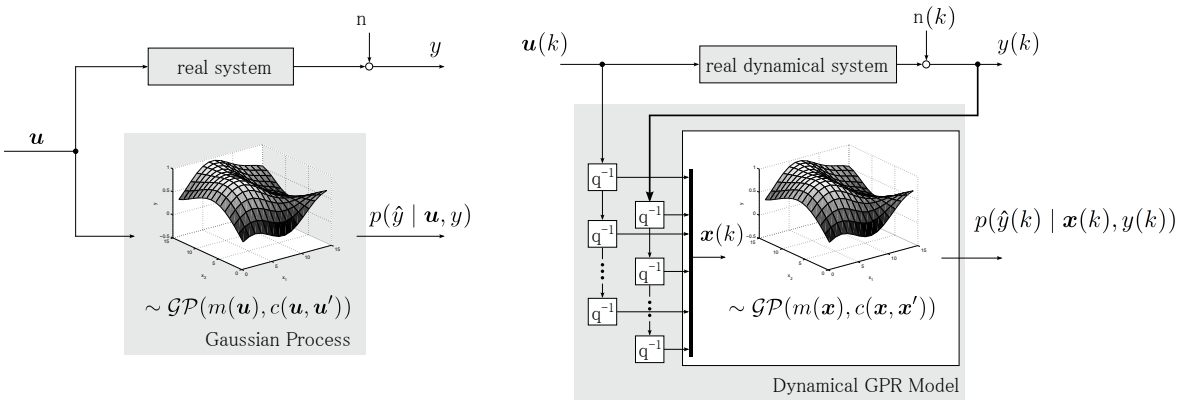


Figure 3.11: GPR model for static and dynamical configuration

Dynamic nonlinear modelling using the GPR algorithm: GPR embodies Bayesian inference and thus *Occam's razor* for an automatic structure optimisation which leads to highly accurate models and provides a high usability. The form of covariance function allows prior knowledge of the system to be defined, which leads to a robust model without a tendancy to overfitting or underfitting. Since GPR structure optimisation is automatic, the predictive variance of the model gives reliable information about the model quality. GPR naturally deals with noisy measurements and unevenly distributed observations (Plagemann et al. 2008). It is appropriate for high-dimensional model learning problems and is thus useful for data-based modelling of dynamical processes, see Gutjahr (2012). For dynamic systems, the extrapolation behaviour leads to weakening effects and stable model behaviour, since in the case of uncertainty, the model output adapts to the mean value of the model. Fig. 3.12 shows a 1D example for the GPR predictive variance.

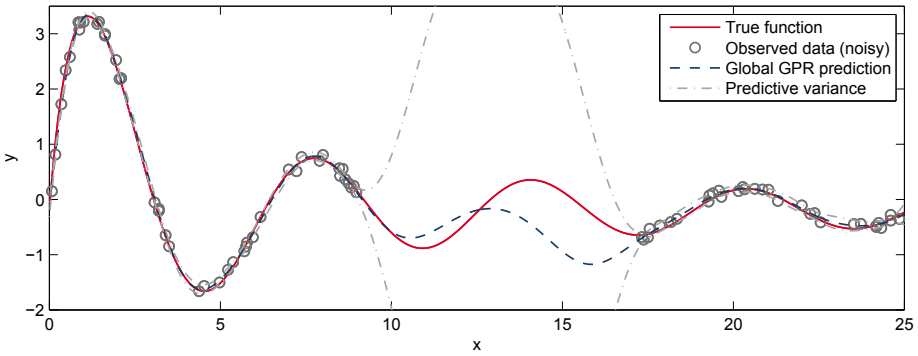


Figure 3.12: Predictive variance of GPR model

The main disadvantages of GPR are caused by the fact that the covariances of all training data have to be stored in the covariance matrix and further be inverted for model training, which yields to slow training speed because of a cost of $\mathcal{O}(n^3)$ (Williams & Rasmussen 1996). To reduce the complexity so-called *Sparse Gaussian Process Regression* (SGPR) methods were developed to approximate the GPR model, see Quiñonero-Candela & Rasmussen (2005) for an overview of SGPR approaches. However, standard GPR is limited to finite training data and not appropriate for large data sets. The GPR model is a global model and thus not well-suited for iterative modelling or online adaptation. Since all training points are used to evaluate the covariance matrix, the model is in general fixed after model training.

3.2.5 Summary of model comparison

In the last sections the pros and cons of the polynomial model, MLP, LOLIMOT, HILOMOT and GPR were presented, and their suitability for the identification of dynamic nonlinear systems was discussed. The following table summarises the properties of the different model types due to the requirements, presented in section 3.1.

Properties	Polynomial	MLP	LOLIMOT	HILOMOT	GPR
Accuracy	-[1]	+[1]	0[1]	+[1]	++
Parameter estimation	++[1]	--[1]	++[1]	+[1]	--
Structure optimisation	-[1]	_[1]	+[1]	+[1]	++
Sensitivity to noise	+[1]	++[1]	++[1]	+[1]	++
Interpretation	0[1]	--[1]	++[1]	+[1]	++
Incorporation of prior knowledge	-[1]	--[1]	++[1]	+[1]	0
High dimensional mapping	--[1]	+[1]	-/0[1]	0/+	++
Large data sets	++	0	+	0	--
Training speed	+[1]	--[1]	++[1]	+[1]	-
Uneven data distribution	--	++	n.n.	n.n.	0
Expression of uncertainty	-	--	0	0	++
Weakening extrapolation	--	0	+	0	++
Iterative modelling	--	--	++	++	--
Model recalibration	--	--	++	++	--
Noise variations	--	--	+	+	--
Adaptation of local complexity	--	++	++	++	--
Adaptation of local changing dynamics	--	--	++	++	--
Simulation speed	0[1]	+[1]	+[1]	0[1]	-
Requirement of memory	++	+	0	+	-
Online adaptation	-[1]	--[1]	++[1]	++[1]	--
Effort for ECU implementation	++	+	+	0	-
Usability	0	-	++	++	++

[1] Nelles (2001)

Given the results of the table, it becomes clear that polynomial models and MLP networks have many disadvantages compared to LOLIMOT, HILOMOT and GPR. Regarding model accuracy, structure optimisation and high dimensional mapping, GPR shows superior properties regarding the regression problem of dynamic nonlinear systems. Furthermore, the reliable expression of uncertainty is mandatory when dealing with dynamic measurements. The drawback of the training speed and the restriction of the dataset size can be circumvented by using SGPR.

The main drawbacks of GPR are more related to the practical usage of dynamic identification. Here the LOLIMOT algorithm stands for flexibility and possibilities for adaptation and further allows online adaptation. However, the performance of LOLIMOT for high dimensional problems is low and leads to inaccurate predictions. Regarding these points, HILOMOT is beneficial but leads to other restrictions.
However, the results of the model comparison motivate the approach of combining the powerful static approximator of GPR with the advantages of the LOLIMOT algorithm. Furthermore, the benefit of the uncertainty output of the GPR model can be used instead of validity functions to combine *Local Gaussian Process Regression* in a consistent way. In the following, the algorithm of Local Gaussian Process Regression will be introduced.

3.3 Theory of Gaussian Process Regression

The preceding sections motivate the usage of GPR for dynamic nonlinear modelling. In this section the theory of GPR is explained in detail. Starting with the related work to GPR, the Bayesian machinery is introduced. The combination of the Bayesian probability theory and the theory of multivariate Gaussian distribution leads to the powerful regression algorithm of Gaussian Process Regression.

3.3.1 Related Work

The framework of Bayesian inference with Gaussian Processes for machine learning purposes was first presented in Williams (1995) and in Williams & Rasmussen (1996). Based on the research of MacKay (1992*a*) and Neal (1995) on Bayesian inference with neural networks, Williams and Rasmussen investigated a more practical way to use the Bayesian framework to solve regression problems. By defining a stochastic process as a prior over functions, which was first introduced by O'Hagan (1978), Williams' and Rasmussen's investigation permitted the predictive Bayesian analysis for fixed values of hyperparameters to be carried out exactly using matrix operations (Williams & Rasmussen 1996). For a comprehensive introduction to GPR, see Gibbs & MacKay (1997), Williams & Rasmussen (2002), Rasmussen & Williams (2006).

3.3.2 Bayesian framework

The *Bayesian* framework is distinguished by its use of probability to express all forms of uncertainty. By using rules of probability Bayesian learning can be performed to express in terms of a probability distributions over all unknown quantities (Neal 1995). Bayes gives a direct answer about probabilities and thus about uncertainties. Furthermore, this approach allows to put prior beliefs into the model which avoids overfitting. The next benefit of the Bayesian method is that it automatically and quantitatively embodies *Occam's razor* without the introduction of ad hoc penalty terms and thus, complex models are automatically self-penalizing under Bayes' rule (MacKay 1992*b*). An excellent introduction to the Bayesian framework can be found in Jaynes (1986) and Loredo (1989). The Bayesian framework is fundamental for Gaussian Process Regression and thus in this thesis the basics are introduced in a similar way to Loredo (1989).
Given the *'Product rule'* of probability theory by

$$\boxed{p(AB \mid C) = p(A \mid BC) \times p(B \mid C)} \tag{3.6}$$

and regarding the fact, that AB is identical to BA, then Eq. 3.6 implies, that

$$p(A \mid BC) \times p(B \mid C) = p(B \mid AC) \times p(A \mid C) \tag{3.7}$$

and solving Eq. (3.7) for $p(A \mid BC)$ gives us what we know today as *'Bayes' Theorem'*, formulated by Bayes and presented in a general way by Laplace 1774 (Loredo 1989):

$$\boxed{p(A \mid BC) = \frac{p(B \mid AC) \times p(A \mid C)}{p(B \mid C)}} \tag{3.8}$$

where A,B,C denotes various propositions, AB stands for 'A and B are true' and $p(A \mid B)$ stands for the 'probability that A is true, given that B is true' (Jaynes 1986). Although Eq. (3.8) is a trivial result of the Product Rule, it embodies a mathematical representation of the *process of learning* since it implements our beliefs of A, although we only know C (Jaynes 1986). Thus, $p(A \mid C)$ is called *prior probability*. The *posterior probability* $p(A \mid BC)$ will be updated as a result of acquiring new information B. In terms of regression, A represents a hypothesis about the model, B represents the data from the observation of the process, and C represents what we know about A before getting the data B (Jaynes 1986). Thus, the posterior probability A is obtained by multiplying our prior probability $p(A \mid C)$ by the probability of the data assuming

the truth of the hypothesis, $p(B \mid AC)$, and dividing by the probability that we would have seen the data anyway, $p(B \mid C)$ (Loredo 1989). In the context of regression, the hypothesis $\mathcal{L}(A)$ is considered as a function, thus $p(B \mid AC)$ is called the *likelihood function* (Loredo 1989).
In the following sections, the general Bayes' rule will be presented in the context of regression, where it is generally called *Bayesian modelling* or *Bayesian learning*.

3.3.2.1 Bayesian learning - posterior and predictive distribution

Now a probabilistic model for regression shall be considered where some quantities $y^{(1)}$, $y^{(2)}$, ..., were generated by some system and every $y^{(i)}$ is independent and identically distributed (i.i.d). y is measured and affected by some additive noise, e.g. sensor noise (see Fig. 3.1). The model has to embody this assumption of noise and thus the general regression model can be formulated as:

$$\boxed{y = f(\boldsymbol{x}) + \varepsilon} \tag{3.9}$$

Here, the model value $f(\boldsymbol{x})$ is added by a noise term ε, which can, for example, be modelled as an independent Gaussian distribution with zero mean and variance σ_n^2.

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$$

The assumption of Gaussian noise is appropriate when dealing with the Bayesian probability theory whenever it is known or considered that the noise has zero mean and finite standard deviation without knowing further details, see Jaynes (1985), Bretthorst (1988) and Bretthorst (1990). Generally, the model $f(\boldsymbol{x})$ consists of parameters $\boldsymbol{w}$ which determine the probability distributions of the $y^{(i)}$. Such probabilities, or probability densities, will be written as $p(y^{(i)} \mid \boldsymbol{w})$ Neal (1995). As mentioned in the previous section the modelling process can be distinguished in two levels of inference. The first is called *estimation*, where a fix model or hypothesis $\mathcal{H}$ with a set of parameters $\boldsymbol{w}$ is assumed, asserting that one of the possible parameter values is true value to model the data $\mathcal{D}$. Given this configuration, Bayes' Rule of Eq. (3.8) now is given as

$$\overbrace{p(\boldsymbol{w} \mid \mathcal{D}\mathcal{H})}^{Posterior} = \frac{\overbrace{p(\mathcal{D} \mid \boldsymbol{w}\mathcal{H})}^{Likelihood} \times \overbrace{p(\boldsymbol{w} \mid \mathcal{H})}^{Prior}}{\underbrace{p(\mathcal{D} \mid \mathcal{H})}_{Evidence}} \tag{3.10}$$

Bayesian learning means updating the prior beliefs, given as a prior distribution, to a posterior distribution by observing data $\mathcal{D}$. Thereby the prior of the parameters depends on their function within the model and so the distribution of $\boldsymbol{w}$ is conditioned on the hypothesis $\mathcal{H}$. Now, data $\mathcal{D}$ arrives from the observation and the term $p(\mathcal{D} \mid \boldsymbol{w}\mathcal{H})$ expresses how likely the data is given that it was generated by the model $\mathcal{H}$ with a specific set of parameters $\boldsymbol{w}$ (Gibbs 1997). Since the fixed model $\mathcal{H}$ is assumed for the step of estimation and regarding a regression problem with real valued targets, where $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, ..., n\} = (\boldsymbol{X}, \boldsymbol{y})$, Eq. (3.10) can be modified to

$$p(\boldsymbol{w} \mid \boldsymbol{y}\ \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}\ \boldsymbol{w}) \times p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{X})} \tag{3.11}$$

Assuming the noise term in Eq. (3.9) is Gaussian, the probability density of the observations given the data and the parameters (likelihood) is:

$$p(\boldsymbol{y} \mid \boldsymbol{X}\ \boldsymbol{w}) = \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i\ \boldsymbol{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - f(x_i))^2}{2\sigma_n^2}\right) \tag{3.12}$$

Since the distribution of $\boldsymbol{w}$ is desired, the *Evidence* term in Eq. (3.11), which for estimation just plays a role of a normalization constant, can be ignored in the first step of inference and thus the combination of likelihood and prior can be formulated as proportional to the posterior:

$$\overbrace{p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X})}^{Posterior} \propto \overbrace{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w})}^{Likelihood} \times \overbrace{p(\boldsymbol{w})}^{Prior} \tag{3.13}$$

Now the posterior distribution of the parameters can be computed by defining the model $f(x)$ to compute the likelihood in Eq. (3.12) and the prior distribution for the parameters $p(\boldsymbol{w})$. The data $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, ..., n\}$ which are used to find the posterior are called *training data*. Given the posterior distribution of the parameters, we want to make predictions given the model $\mathcal{H}$ and the proper parameters $\boldsymbol{w}$ at a query point $\boldsymbol{x}_* \notin \mathcal{D}$. These query points were not part of the training process and thus they are called *test data*. Since the probability theory has been used so far, it is possible to predict the probability distributions of the model value f_*. This is called the *predictive distribution*, sometimes also called *posterior predictive distribution*. The

predictive distribution can be computed by *marginalisation* over the parameter $\boldsymbol{w}$ and using Product Rule of Eq. (3.6):

$$\begin{aligned} p(f_* \mid \boldsymbol{x}_*, \mathcal{D}) &= \int p(f_*, \boldsymbol{w} \mid \boldsymbol{x}_*, \mathcal{D}) \; d\boldsymbol{w} \\ &= \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}, \mathcal{D}) \times p(\boldsymbol{w} \mid \boldsymbol{x}_*, \mathcal{D}) \; d\boldsymbol{w} \end{aligned}$$

Since the f_* is conditioned on x_* but conditionally independent of the data $\mathcal{D}$ and $\boldsymbol{w}$ conditionally independent of the query point $\boldsymbol{x}_*$, the predictive distribution now reads:

$$\overbrace{p(f_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}^{Predictive} = \int \overbrace{p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w})}^{Likelihood_*} \times \overbrace{p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y})}^{Posterior} \; d\boldsymbol{w} \tag{3.14}$$

The predictive distribution states that the probability of f_* is just its average likelihood, taking the average over the posterior distributions for $\boldsymbol{w}$ based on the observed data $\mathcal{D}$ (Loredo 1989). Producing such a distribution is a big advantage compared to other regression methods, since the predictive distribution gives information about the uncertainty of the model. This property will be the key step of the extended algorithm to Gaussian Process Regression, which will be presented in section 3.4.1.

3.3.2.2 Bayesian learning - Example

In Rasmussen & Williams (2006) a simple example for the standard linear regression model is given. This example will be used to get a deeper understanding of the standard method of Bayesian learning. The following model is considered:

$$f(x) = \boldsymbol{x}^T \boldsymbol{w} \quad , \quad y = f(\boldsymbol{x}) + \varepsilon \quad , \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2)$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ a vector of parameters of the linear model, $f(\boldsymbol{x})$ is the function value and y the observed target value. The difference between y and $f(\boldsymbol{x})$ is given as additive noise and further as i.i.d. Gaussian with zero mean and variance σ_n^2. Now the model is defined, the posterior distribution of Eq. (3.13) can be computed by defining the prior distribution of the parameters $p(\boldsymbol{w})$, which expresses the beliefs about the parameters before seeing any data. A multivariate Gaussian with zero mean and covariance matrix Σ_p is chosen to describe the prior $p(\boldsymbol{w})$.

$$\boldsymbol{w} \sim \mathcal{N}(0, \Sigma_p)$$

The density function of the multivariate Gaussian prior is given as:

$$p(\boldsymbol{w}) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_p|}} \exp\left(-\frac{1}{2}\boldsymbol{w}^T \Sigma_p^{-1} \boldsymbol{w}\right)$$

where $|\Sigma_p|$ is the determinant of Σ_p and n the dimension of the parameters. Given the model for $f(x)$ the likelihood of Eq. (3.12) now reads:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{X}\ \boldsymbol{w}) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\left(-\frac{(y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\left(-\frac{1}{2\sigma_n^2}|\boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{w}|^2\right) \\ &= \mathcal{N}(\boldsymbol{X}^T \boldsymbol{w}, \sigma_n^2 \boldsymbol{I}) \end{aligned}$$

and it can be seen that the likelihood is also multivariate Gaussian. Now the posterior distribution of Eq. (3.13)) can be computed as:

$$\begin{aligned} &\textit{Posterior} \propto \textit{Likelihood} \times \textit{Prior} \\ p(\boldsymbol{w} \mid \boldsymbol{y}\ \boldsymbol{X}) &\propto \exp\left(-\frac{1}{2\sigma_n^2}|\boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{w}|^2\right) \exp\left(-\frac{1}{2}\boldsymbol{w}^T \Sigma_p^{-1} \boldsymbol{w}\right) \end{aligned} \tag{3.15}$$

Since a multivariate Gaussian is given as:

$$p(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w} \mid \bar{\boldsymbol{w}}, \Sigma) = \frac{1}{\sqrt{|2\pi\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{w} - \bar{\boldsymbol{w}})^T \Sigma^{-1} (\boldsymbol{w} - \bar{\boldsymbol{w}})\right) \tag{3.16}$$

it becomes obvious that the posterior in Eq. 3.15 is multivariate Gaussian, because it is *quadratic* in $\boldsymbol{w}$. By *'completing the squares'* (see Appendix A.1.2) the posterior is given as:

$$\begin{aligned} p(\boldsymbol{w} \mid \boldsymbol{y}\ \boldsymbol{X}) &\propto \exp\left(-\frac{1}{2\sigma_n^2}(\boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{w})\right) \exp\left(-\frac{1}{2}\boldsymbol{w}^T \Sigma_p^{-1} \boldsymbol{w}\right) \\ &\propto \exp\left(-\frac{1}{2}(\boldsymbol{w} - \bar{\boldsymbol{w}})^T \left(\frac{1}{2\sigma_n^2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1}\right)(\boldsymbol{w} - \bar{\boldsymbol{w}})\right) \end{aligned}$$

where $\bar{\boldsymbol{w}} = \sigma_n^{-2}(\sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1})^{-1}\boldsymbol{X}\boldsymbol{y}$. The posterior distribution is also Gaussian with mean $\bar{\boldsymbol{w}}$ and covariance matrix $\boldsymbol{A}^{-1}$.

$$p(\boldsymbol{w} \mid \boldsymbol{y}\ \boldsymbol{X}) \sim \mathcal{N}\left(\frac{1}{\sigma_n^2}\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{A}^{-1}\right)$$

with $\boldsymbol{A} = \sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1}$. Now proper parameters $\boldsymbol{w}$ are estimated for the fixed model $\mathcal{H}$. Eq. (3.14) delivers the predictive distribution for a query test point $\boldsymbol{x}_*$ as:

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{x}_*^T\boldsymbol{A}^{-1}\boldsymbol{x}_*)$$

3.3.2.3 Bayesian learning - Model Comparison

Section 3.3.2.1 introduced the *first step of inference* for Bayesian learning. By computing the posterior distribution of the parameters, given a fixed model $\mathcal{H}$, it was able to make predictions with the Bayesian model by computing the predictive distribution of the model outputs, given a new query point. However, the *second step of inference* assumes that it is actually not sure that the model $\mathcal{H}$ is the right one for the underlying regression problem and if the model is inadequate, then some alternative model must be better, and so the Bayesian probability theory assesses a model by comparing it to one or more alternatives (Loredo 1989). This step of inference is called *model comparison* or *model selection*. Here the terminology *model* is a bit confusing because we are not looking for different types of models, e.g. polynomial, LOLIMOT etc., we are looking for different complexity levels. Roughly speaking, the complexity means the flexibility of a model. Regarding the example of the polynomial models in section 3.1 the three polynomials are different models $\mathcal{H}$ with their own parameters $\boldsymbol{w}$. Next to the parameters $\boldsymbol{w}$ there are also parameters of $\mathcal{H}$ which control the complexity and are not used to fit the data. If the complexity of the model is high then it will be able to fit the data with the result of a small *bias*, but the *variance* will be high. In contrast, if the model complexity is low, maybe just a line, then the bias is high but the variance is low. This *bias/variance dilemma* was already introduced in section 3.1, see Fig. 3.5. However, in the Bayesian probability theory the complexity controlling parameters are called *hyperparameters*. In the example of section 3.3.2.2 the prior for the distribution of $p(\boldsymbol{w})$ and also the noise parameter σ_n were hyperparameters of the model. As the requirements for regression pointed out, the best model is not the one which fits the data best but the one that is the most likely to be able to generalise and make predictions for new query points. This chapter will show that the Bayesian solution of *model selection* implements *Occam's razor* which automatically prefers simpler models unless a more complicated model provides a significantly better fit to the data (Loredo 1989). However, the aim is to find the distribution of f_* at $\boldsymbol{x}_*$, given the parameters $\boldsymbol{w}$ for a model $\mathcal{H}$ consisting of hyperparameters controlling the flexibility

of the model. Therefore the predictive distribution of Eq. 3.14 can be rewritten as a marginalisation of the model $\mathcal{H}$ and using Product Rule of probability of Eq. 3.6:

$$\begin{aligned} p(f_* \mid \boldsymbol{x}_*, \mathcal{D}) &= \int p(f_*, \mathcal{H} \mid \boldsymbol{x}_*, \mathcal{D}) \; d\mathcal{H} \\ &= \int p(f_* \mid \boldsymbol{x}_*, \mathcal{D}, \mathcal{H}) \times p(\mathcal{H} \mid \boldsymbol{x}_*, \mathcal{D}) \; d\mathcal{H} \end{aligned}$$

Since $\mathcal{H}$ is conditionally independent of the query point $\boldsymbol{x}_*$ ($\boldsymbol{x}_*$ is not random), and f_* independent of the data $\mathcal{D}$, given the parameters $\boldsymbol{w}$, the predictive distribution is given as:

$$\begin{aligned} p(f_* \mid \boldsymbol{x}_*, \mathcal{D}) &= \int p(f_* \mid \boldsymbol{x}_*, \mathcal{D}, \mathcal{H}) \times p(\mathcal{H} \mid \mathcal{D}) \; d\mathcal{H} \\ &= \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}, \mathcal{H}) \times p(\mathcal{H} \mid \mathcal{D}) \; d\mathcal{H} \end{aligned} \tag{3.17}$$

and $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}, \mathcal{H})$ was already computed in Eq. 3.14 for the predictive distribution by marginalisation of the parameter $\boldsymbol{w}$:

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}, \mathcal{H}) = \int p(f_* \mid \boldsymbol{x}_*, \boldsymbol{w}, \mathcal{H}) \times p(\boldsymbol{w} \mid, \mathcal{D}, \mathcal{H}) \; d\boldsymbol{w} \tag{3.18}$$

It can be seen in Eq. 3.17 and Eq. 3.18 that using Bayesian machinery means averaging over the probabilities of f_* which are predicted by the different models $\mathcal{H}$ and weighted by the posterior distributions of the models $\mathcal{H}$, given the data $\mathcal{D}$. This is called *Bayesian Model Averaging*. However, for complex model types the integral in Eq. 3.17 is hard to compute and so an approximation of $p(f_* \mid \boldsymbol{x}_*, \mathcal{D})$ is useful. Thus a point estimation of the model is used.

$$p(f_* \mid \boldsymbol{x}_*, \mathcal{D}) \approx p(f_* \mid \boldsymbol{x}_*, \mathcal{D}, \mathcal{H}^*) \quad \text{where} \quad \mathcal{H}^* \in \arg\max_m p(\mathcal{H} \mid \mathcal{D})$$

Here, $\mathcal{H}^*$ is a maximum a-postiori estimate of the model. Given Bayes' Rule of Eq. 3.8, $p(\mathcal{H} \mid \mathcal{D})$ can be written as:

$$p(\mathcal{H} \mid \mathcal{D}) \propto \overbrace{p(\mathcal{D} \mid \mathcal{H})}^{Evidence} \times p(\mathcal{H}) \tag{3.19}$$

It can also be seen that the posterior of the models $p(\mathcal{H} \mid \mathcal{D})$ is proportional to the so-called *Marginal Likelihood* times the prior of the models and if a uniform distribution of the models is assumed then the maximum a-posteriori of the models can be computed by the maximum of the marginal likelihood. To compute the marginal likelihood Bayes' Rule of Eq. 3.8 has to be recalled. Since it is known that true parameters exist,

all possible parameters $\boldsymbol{w}_1 + \boldsymbol{w}_2 + ...$ given the model is true and thus $p(\boldsymbol{w}_1 + \boldsymbol{w}_2 + ... \mid \mathcal{H}) = 1$ and $p(\mathcal{D} \mid \mathcal{H})$ can be written as

$$\begin{aligned} p(\mathcal{D} \mid \mathcal{H}) &= p(\mathcal{D} \mid \mathcal{H}) \times p(\boldsymbol{w}_1 + \boldsymbol{w}_2 + ... \mid \mathcal{H}) \\ &= p(\mathcal{D}\ [\boldsymbol{w}_1 + \boldsymbol{w}_2 + ...] \mid \mathcal{H}) \end{aligned}$$

Expanding this term using the Product Rule of probabilities (Eq. 3.6) gives:

$$\begin{aligned} p(\mathcal{D}\ [\boldsymbol{w}_1 + \boldsymbol{w}_2 + ...] \mid \mathcal{H}) &= p(\mathcal{D}\ \boldsymbol{w}_1 \mid \mathcal{H}) + p(\mathcal{D}\ \boldsymbol{w}_2 \mid \mathcal{H}) + ... \\ &= \sum_k p(\mathcal{D}\ \boldsymbol{w}_k \mid \mathcal{H}) \\ &= \sum_k p(\boldsymbol{w}_k \mid \mathcal{H}) \times p(\mathcal{D} \mid \boldsymbol{w}_k\ \mathcal{H}) \end{aligned}$$

and together

$$p(\mathcal{D} \mid \mathcal{H}) = \sum_k p(\boldsymbol{w}_k \mid \mathcal{H}) \times p(\mathcal{D} \mid \boldsymbol{w}_k\ \mathcal{H})$$

which for continuous parameters gives an integral as:

$$\boxed{\text{Evidence:} \quad p(\mathcal{D} \mid \mathcal{H}) = \int p(\boldsymbol{w} \mid \mathcal{H}) \times p(\mathcal{D} \mid \boldsymbol{w}\ \mathcal{H}) d\boldsymbol{w}}$$

and the name *marginal likelihood* comes from the fact, that we marginalise over the parameters $\boldsymbol{w}$, see Loredo (1989). In general: to assign a preference to alternative models $\mathcal{H}$, independent of parametric or non-parametric models, a Bayesian evaluates the *evidence* (marginal likelihood) $p(\mathcal{D} \mid \mathcal{H})$ (MacKay 1992*b*). But where actually is Occam's razor which regulates the complexity and prefers simpler models to complex ones? The *evidence* naturally embodies Occam's razor:
MacKay (1992*b*) mentioned that for many problems it is common for the posterior $p(\boldsymbol{w} \mid \mathcal{D}, \mathcal{H}) \propto p(\mathcal{D} \mid \boldsymbol{w}\mathcal{H}) \times p(\boldsymbol{w} \mid \mathcal{H})$ to have a strong peak at the most probable parameters $\boldsymbol{w}_{MP}$, see Fig. 3.13

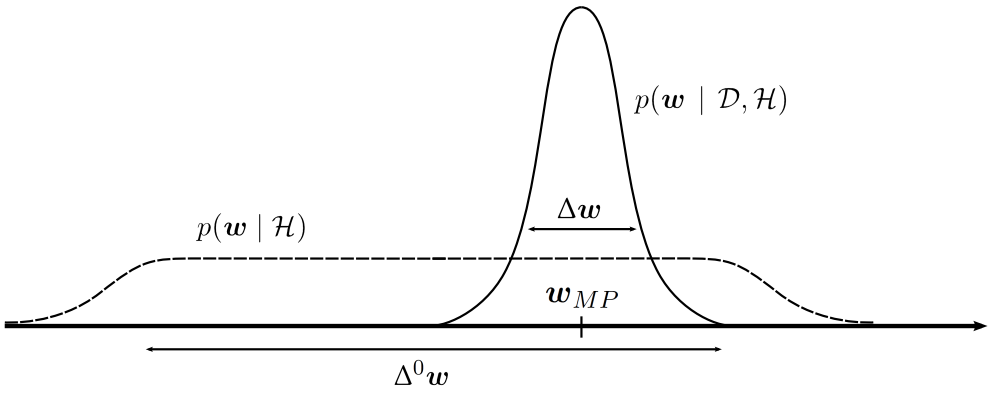


Figure 3.13: The Occam factor $\Delta \boldsymbol{w} / \Delta^0 \boldsymbol{w}$ (MacKay 1992*b*)

Thus the evidence can be approximated by the height of the peak of the integrand $p(\boldsymbol{w} \mid \mathcal{H}) \times p(\mathcal{D} \mid \boldsymbol{w}\ \mathcal{H})$ times its width, $\Delta \boldsymbol{w}$:

$$\underbrace{p(\mathcal{D} \mid \mathcal{H})}_{Evidence} \approx \underbrace{p(\mathcal{D} \mid \boldsymbol{w}_{MP}\ \mathcal{H})}_{Best fit\ likelihood} \times \underbrace{p(\boldsymbol{w}_{MP} \mid \mathcal{H}) \Delta \boldsymbol{w}}_{Occam factor}$$

Generally, the evidence is found by taking the best fit likelihood that the model can achieve and multiplying it by the Occam factor (Gull 1988). The interpretation of the Occam factor is visualised in Fig. 3.13. Here the posterior uncertainty in $\boldsymbol{w}$ is expressed by $\Delta \boldsymbol{w}$ and $\Delta^0 \boldsymbol{w}$ expresses the width of the prior distribution. In other words, complex models with many parameters will be penalised with a high Occam factor, because of the large range of $\Delta^0 \boldsymbol{w}$.

3.3.2.4 Bayesian learning - Overview and Criticism

In the preceding sections the Bayesian framework was introduced and a simple example was given for a linear regression model. It was shown that the key idea of Bayesian learning is to put probability distributions on all unknown quantities and to use the rules of probability in order to update prior beliefs to a posterior distribution by observing data. Using probabilities and prior beliefs is beneficial to other modelling approaches since the model predictions can also be expressed by probability distributions and thus information about the uncertainty of the model is given. Fig. 3.14 shows an overview of Bayesian learning for automotive identification. The two steps of inference as shown receive the most probable parameters $\boldsymbol{w}$ and the most probable model $\mathcal{H}$.

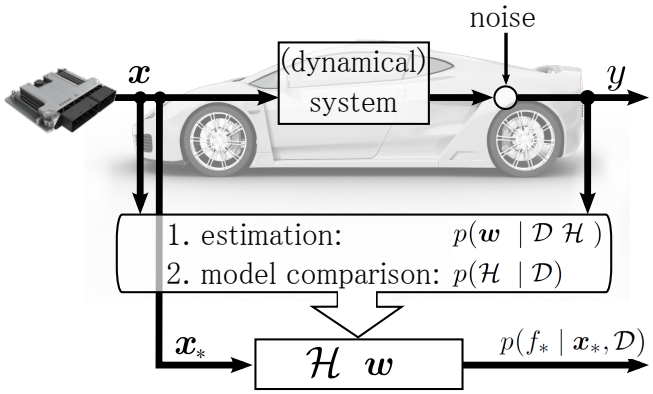


Figure 3.14: Overview of Bayesian learning

Although Bayesian learning has the mentioned benefits it is often not easy to use it in practise. One criticism of this approach is that defining priors for the model parameters of complex models seems to be obscure when the model parameters have no interpretability. Furthermore, defining the model itself is a crucial step towards being successful with the Bayesian machinery. Nevertheless, solving the integral for the predictive distribution of Eq. 3.14 is very difficult. However, MacKay (1992*a*) used the Bayesian approach for simple Neural Networks by defining Gaussian distributions over the weights and biases. Neal (1995) showed that the Bayesian approach can also be successfully applied to complex models. In his Ph.D. thesis he showed that there is no need to limit the number of hidden units and that a Gaussian prior for hidden-to-output weights results in a Gaussian process for functions that may be smooth, for example (Neal 1995). However, the prior over functions has a complex form and thus the computation requires approximations (MacKay 1992*a*) or Monte Carlo approaches to evaluate integrals (Neal 1995). Inspired by the work of Neal on priors for infinite networks, Williams & Rasmussen (1996) published a new method for machine learning purposes using Gaussian Processes. This approach will be discussed in the next section.

3.3.3 Gaussian Process Regression

In the preceding chapter the advantages of Bayesian probability theory were explained but also the criticism of practical usage relating to complex models. The combination of the Bayesian probability theory with complex models was introduced by MacKay (1992*a*) and Neal (1995) and pursued by Williams & Rasmussen (1996) whose approach

of Gaussian Process Regression provides simple settings of priors for GP instead of the difficult task of setting priors over a large number of network weights (Rasmussen 1996). This means that instead of defining a prior over model parameters to fit the data, a prior for the function itself is defined. Since a Gaussian Processes is used, the predictive distribution can be computed easily by conditioning, using rules for multivariate Gaussian distributions. In this section GP is introduced and how to make predictions with it. It will be shown how to adapt the hyperparameters of the covariance function to achieve the powerful Gaussian Process Regression algorithm.

3.3.3.1 Definition of a Gaussian Process

In general, a *stochastic process* is a collection of random variables $\{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\}$ indexed by a set $\boldsymbol{X}$ which is the input space of dimension d, in the case of regression, the number of inputs. The stochastic process is specified by giving the probability distribution for every finite subset of variables $f(\boldsymbol{x}^{(1)}), ..., f(\boldsymbol{x}^{(k)})$ in a consistent manner (Williams & Rasmussen 1996). The *Gaussian process* is a stochastic process which can be fully specified by its *mean function* $m(\boldsymbol{x})$ and *covariance function* $c(\boldsymbol{x}, \boldsymbol{x'})$ with

$$m(\boldsymbol{x}) = E[f(\boldsymbol{x})]$$
$$c(\boldsymbol{x}, \boldsymbol{x'}) = E[(f(\boldsymbol{x}) - m(\boldsymbol{x}))(f(\boldsymbol{x'}) - m(\boldsymbol{x'}))]$$

any finite set of points will have a joint multivariate Gaussian distribution (Williams & Rasmussen 1996). In the following, GPs with zero mean are considered and thus the properties of the covariance function are the only degree of freedom.
For the regression problem the measurement data is given as: $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}, i = 1, ...n\}$ assuming that any set of $y^{(i)}$ has multivariate Gaussian distribution. Thus the general model $f(\boldsymbol{x})$ can be defined as a Gaussian Process, written as (Rasmussen & Williams 2006):

$$\boxed{f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), c(\boldsymbol{x}, \boldsymbol{x'}))}$$

The GP consists of the covariance function $c(\boldsymbol{x}, \boldsymbol{x'})$ but for a real application we are actually dealing just with a finite set of data, and so the final model will be given as a multivariate Gaussian distribution represented by a *covariance matrix* $C(\boldsymbol{x}, \boldsymbol{x'})$ whose entries describe the covariance between the outputs, given two inputs of the training points (Rasmussen 1996).

3.3.3.2 Predictions with Gaussian Processes and Bayes' Rule

In regression we are interested in making predictions for a new query point $f(\boldsymbol{x}_*)$ and we are further interested in the probability of the predicted model value $p(f_* \mid \boldsymbol{x}_*, \mathcal{D})$ for a set of n training points $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)}) \mid \boldsymbol{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}, i = 1, ...n\}$. Given the assumption of observed values $\boldsymbol{y}$ to be noise corrupted versions of the outputs $f(\boldsymbol{x})$ by a noise term ε we can write:

$$y = f(\boldsymbol{x}) + \varepsilon \quad , \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{3.20}$$

In order to make predictions, we are first interested in the posterior distribution of the model values $\boldsymbol{f}$, which are given using Bayes' Rule of Eq. 3.8:

$$\boxed{p(\boldsymbol{f} \mid \boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X}) \times p(\boldsymbol{f} \mid \boldsymbol{X})}{p(\boldsymbol{y} \mid \boldsymbol{X})} = \frac{likelihood \times prior}{evidence}} \tag{3.21}$$

In contrast to the example of Bayesian learning for parametric models in section 3.3.2.2, we will leave out any parameters and use a non-parametric approach by defining the model values $f^{(1)}, ..., f^{(n)}, f_*$ as stochastic variables modelling the function at the corresponding inputs $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}, \boldsymbol{x}_*$. We want to implement the GP, which can be seen as distribution over function, and thus we assign a multivariate Gaussian distribution to these variables:

$$p(\boldsymbol{f} \mid \boldsymbol{X}) = \mathcal{N}(0, \boldsymbol{K}) \tag{3.22}$$

where $\boldsymbol{K}$ is the covariance matrix, which was introduced in the preceding section. Since we have not considered the true noise affected values $\boldsymbol{y}$, the joint distribution of $\boldsymbol{f}$ can be treated as a prior. This means that we can put our prior beliefs of what we think may lead to a high covariance between the outputs into the covariance matrix $\boldsymbol{K}$ by defining the appropriate covariance function for our beliefs. In an automotive context, often the aim is to model physical systems which are generally *smooth*. Thus, a covariance function is useful which embodies the following property: points which are *close* together in input space are strongly *correlated* and hence give rise to similar values of $\boldsymbol{y}$ (Gibbs & MacKay 1997). The so-called *Squared Exponential Covariance Function* has shown to be appropriate:

$$c(\boldsymbol{x}_p, \boldsymbol{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{j=1}^{d}\frac{(x_{pj} - x_{qj})^2}{w_j^2}\right) \tag{3.23}$$

Inside the covariance function, parameters control the complexity of the model and are thus called *hyperparameters*. For the squared exponential covariance function the *signal variance* σ_f and the *lengthscale* $\boldsymbol{w}$ are hyperparameters. However, we will concentrate on the covariance function and the hyperparameters in the next section. In this section we define the covariance function and the hyperparameters as given and so we can plot samples from the prior distribution. Fig. 3.15 shows examples from the GP prior for different lengthscales and signal variance.

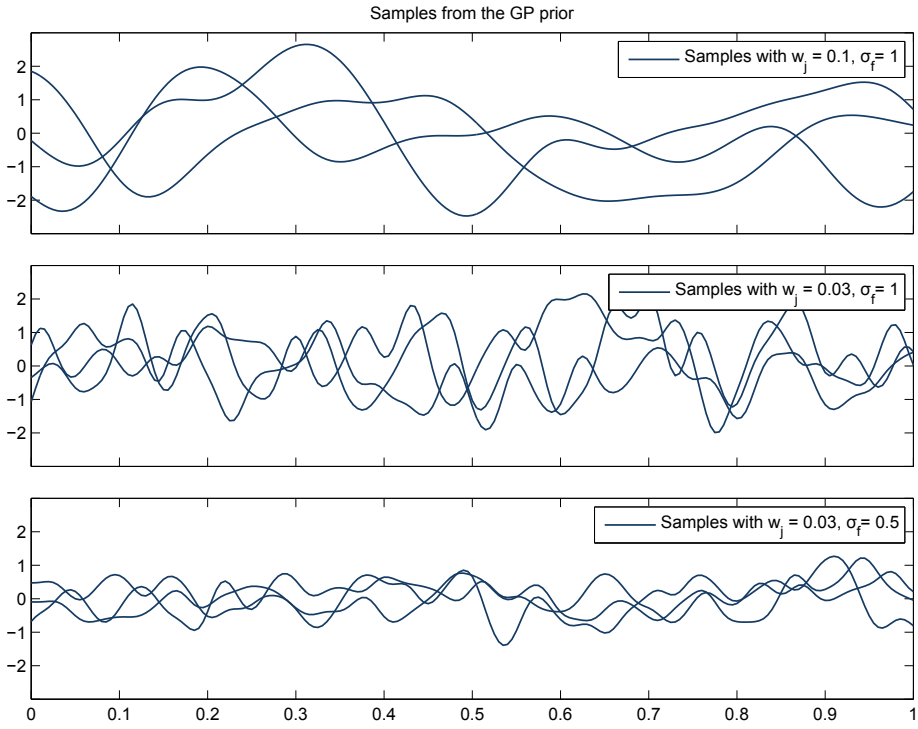


Figure 3.15: Samples from a GP prior with squared exponential covariance function

The probability that the noisy data $\boldsymbol{y}$ appears, given the model values $\boldsymbol{f}$, can also be expressed by a multivariate Gaussian, since we assumed the noise to be independent Gaussian. Thus we receive the likelihood $p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X})$

$$p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{f}, \sigma_n^2 \boldsymbol{I}) \tag{3.24}$$

Now the numerator of Eq. 3.21 can be computed by using the product rule for two Gaussian distributions, see Eq. A.5. The last term is called *evidence* or *marginal*

likelihood, which can be achieved by marginalising over the latent variable f, see Eq. A.8:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{X}) &= \int p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X}) \times p(\boldsymbol{f} \mid \boldsymbol{X}) \; d\boldsymbol{f} \\ &= \int \mathcal{N}(\boldsymbol{f}, \sigma_n^2 \boldsymbol{I}) \times \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) \; d\boldsymbol{f} \\ &= \mathcal{N}(\boldsymbol{0}, \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}) \end{aligned} \tag{3.25}$$

Given the prior, the likelihood and the evidence, each as multivariate Gaussian, the advantage of using GP becomes clear, since the posterior distribution is also Gaussian (von Mises 1964):

$$p(\boldsymbol{f} \mid \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{K}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \boldsymbol{K} - \boldsymbol{K}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{K}) \tag{3.26}$$

The predictive distribution $p(f_* \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})$ can be computed by integrating over the latent variable $\boldsymbol{f}$ and using the product rule of probability, see Eq. 3.6.

$$\begin{aligned} p(f_* \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) &= \int p(f_*, \boldsymbol{f} \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) \; d\boldsymbol{f} \\ &= \int p(f_* \mid \boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) \times p(\boldsymbol{f} \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) \; d\boldsymbol{f} \end{aligned} \tag{3.27}$$

Since we know that f_* is conditionally independent of $\boldsymbol{y}$, we have no measurement in $\boldsymbol{x}_*$, and we know that $\boldsymbol{f}$ is conditionally independent of $\boldsymbol{x}_*$ because the distribution of $\boldsymbol{f}$ is the posterior (see Eq. 3.26) we can write the predictive distribution as follows:

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) = \int p(f_* \mid \boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) \times p(\boldsymbol{f} \mid \boldsymbol{y}, \boldsymbol{X}) \; d\boldsymbol{f} \tag{3.28}$$

Now $p(f_* \mid \boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X})$ is our prior from Eq. 3.22, conditioned at $\boldsymbol{x}_*$, thus we can write

$$\begin{bmatrix} \boldsymbol{f} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^T & k_{**} \end{bmatrix}\right) \tag{3.29}$$

with covariances $\boldsymbol{k}_* = C(\boldsymbol{X}, \boldsymbol{x}_*)$ and $k_{**} = C(\boldsymbol{x}_*, \boldsymbol{x}_*)$. Using the rules for conditional Gaussian (see Eq. A.12) we can write:

$$p(f_* \mid \boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{f}, k_{**} - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_*) \tag{3.30}$$

Given the posterior distribution of Eq. 3.26, and the conditioned prior of Eq. 3.30,

we can compute the integral of the predictive distribution given in Eq. 3.27 by using Eq. A.8, which again leads to a Gaussian distribution

$$p(f_* \mid \boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(m_*, v_*)$$

with mean

$$\boxed{m_* = \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}} \tag{3.31}$$

and variance

$$\boxed{v_* = k_{**} - \boldsymbol{k}_*^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_*} \tag{3.32}$$

Thus, Eq. 3.31 and Eq. 3.32 are all we need to make predictions using the GP model. Since the training data is inside the covariance matrix $\boldsymbol{K}$ and the entries of $\boldsymbol{K}$ will not change for prediction, we can introduce a prediction vector $\boldsymbol{\alpha} = (\boldsymbol{K} + \sigma_n^2 I)^{-1} \boldsymbol{y}$ and thus we can write $m_* = \boldsymbol{k}_*^T \boldsymbol{\alpha}$.

Example: to visualise the prediction f_* for a query point x_* we assume a simple example with only one training point x_1, y_1. Since the hyperparameters for the model are given beforehand, the covariance function can be evaluated for the query point x_* as shown in Fig. 3.16. By conditioning in y_1 we receive the posterior distribution $p(f_* \mid x_*, y_1, x_1)$ and thus the predicted f_*.

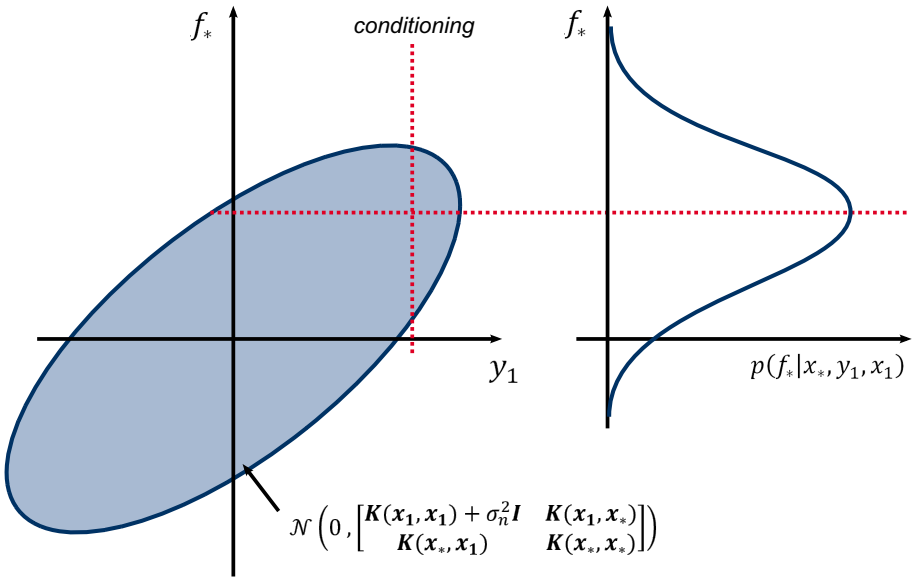


Figure 3.16: GP prediction by conditioning

The predictive variance of Eq. 3.32 can be interpreted as model uncertainty. The

larger the variance, the more uncertain the GPR model is about the prediction for the query point (see Fig. 3.17). This variance will play a key role for the Local Gaussian Process Regression algorithm, which will be presented in chapter 3.4.1.

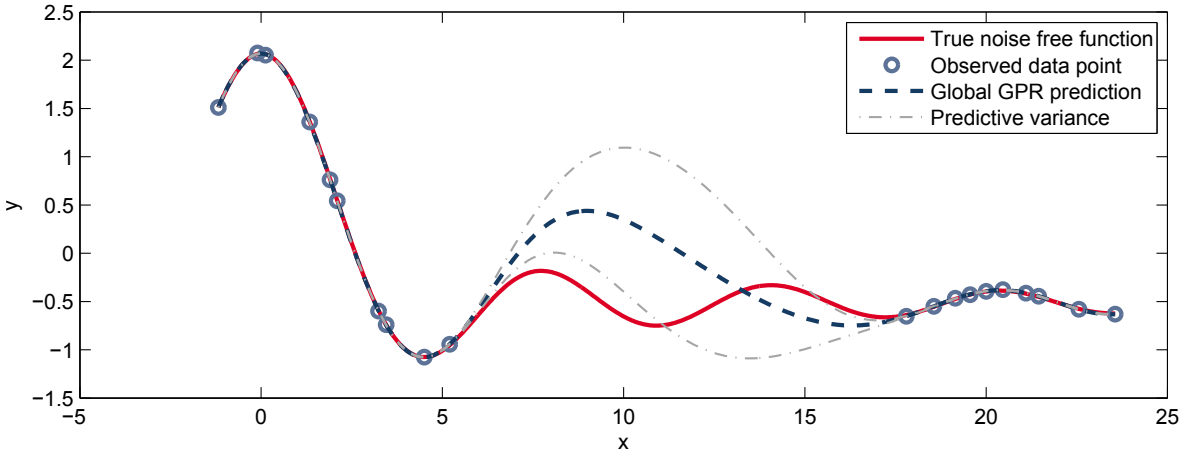


Figure 3.17: Example of a GPR model with predictive variance

3.3.3.3 Adapting the Hyperparameters of the Covariance Function

In the preceding section the GPR model and how to compute the predictive distribution by conditioning it for query points was presented. It was shown, that the crucial ingredient of GPR is the covariance function. The squared exponential covariance function was introduced to be appropriate as it defines a high covariance in the outputs, when the input values are close together. However, from section 3.3.2 we know that we have two steps of inference: one to fit the data and one to compare different model complexities. The predictive distribution of Eq. 3.28 implements the posterior distribution of the model values, given the data, and thus stands for the first step of inference. So far, we have not checked if the model we are using is adequate or if a better one exists. Thus we need the second step of inference: the model comparison. The models that we want to compare are not different in their structure, but different in their complexity, which is controlled by the hyperparameters. Since the covariance function is the only degree of freedom, the hyperparameters are inside the covariance function. Given the squared exponential covariance function of Eq. 3.23 and the assumption of the noise term in Eq. 3.20 we can write the hyperparameters as $\boldsymbol{\theta} = \{\sigma_n, \sigma_f, \boldsymbol{w}\}$. Thus the GPR model consists of one noise variance hyperparameter, one signal variance parameter and for each input one lengthscale hyperparameter, so the parameters scale with $d + 2$, where d is the dimension of inputs. Estimating these

hyperparameters from data by optimisation is called *model training*. The optimisation task is to find the optimal hyperparameters so that the probability for the occurrence of the data $\mathcal{D}$, given the model is maximal. In section 3.3.2.3 the Bayesian model selection was introduced and it was shown that the marginal likelihood or evidence is proportional to the posterior of the model (see Eq. 3.19). The maximum a-posteriori of the models is thus given by maximising the evidence, or more efficiently by minimising the negative marginal log-likelihood (Rasmussen & Williams 2006):

$$\varphi = -log(p(\boldsymbol{y} \mid \boldsymbol{X}, \theta)) \rightarrow \min_{\theta}$$

Here, the marginal likelihood can be computed by marginalisation of the joint distribution of the likelihood $p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X})$ and the prior $p(\boldsymbol{f} \mid \boldsymbol{X})$ as described in section 3.3.3.2.

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X}) \times p(\boldsymbol{f} \mid \boldsymbol{X})\, d\boldsymbol{f}$$

We have already formulated the marginal likelihood to compute the posterior, see Eq. (3.25) and received a multivariate Gaussian with dimension n as:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta}) &= \mathcal{N}(0, \boldsymbol{K} + \sigma_n^2 \boldsymbol{I}) \\ &= \frac{1}{(2\pi)^{n/2} |\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}|^{1/2}} \exp\left(-\frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}\right) \end{aligned}$$

for numerical reasons we use the logarithm:

$$\begin{aligned} \log(p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})) &= \log\left(\frac{1}{(2\pi)^{n/2} |\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}|^{1/2}}\right) - \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ &= \log(1) - \log((2\pi)^{n/2} |\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}|^{1/2}) - \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ &= -\log((2\pi)^{n/2}) - \log(|\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}|^{1/2}) - \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y} \\ &= -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log(|\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}|) - \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y} \end{aligned}$$

and thus finally the negative logarithm of the marginal likelihood is given by

$$\boxed{\varphi = -log(p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{\theta})) = \frac{1}{2} log \mid \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} \mid + \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y} + \frac{n}{2} log(2\pi)} \tag{3.33}$$

In this equation, the first term $\frac{1}{2} log \mid \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} \mid$ is a complexity penalty term, which measures and penalises the complexity of the model. The second term $\frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}$ is dependent on the training set output values $\boldsymbol{y}$ and plays the role of a

data-fit measure. Given these terms, the optimal GP model is a tradeoff between the penalty of the model complexity and data-fit. The tradeoff between data-fit and model complexity is automatic: this effect is called Occam's razor (Nelles 2001, Nguyen-Tuong et al. 2009). To optimise the hyperparameters $\boldsymbol{\theta}$ a multivariate optimisation can be employed. Here, the method of *conjugate gradients* is a common choice, see Rasmussen (1996) for detailed information.

3.4 Variance Weighted Local Gaussian Process Regression

Motivated by the model comparison in chapter 3.1 the GPR algorithm was introduced in the preceding section. As the model comparison pointed out, standard GPR shows disadvantages due to dynamic modelling for practical usage. These properties will now be discussed and used to motivate the *Variance Weighted Local Gaussian Process Regression* (VW-LGPR), presented in Tietze et al. (2014b). This chapter closely follows the proposals in Tietze et al. (2014b).

- *Large data sets and Training performance*: The major limitation of GPR is the expensive computation of the inverse matrix $(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1}$ for model training (see Eq. 3.33) which yields a cost of $\mathcal{O}(n^3)$ (Williams & Rasmussen 1996). The *squared exponential kernel*, given in Eq. 3.23, implements the term $\|\boldsymbol{x} - \boldsymbol{x}'\|$ i.e. that the covariance between function values $f(\boldsymbol{x})$ and $f(\boldsymbol{x}')$ is a function of *Euclidean distance*. This property is called *stationarity*. The entries of the covariance matrix describe the correlation of the model outputs, based on the Euclidean distance of the input values. The correlation of two model values, whose input values are far away from each other, will thus be very small. A simple example will give an intuitive understanding of the covariance matrix. Given unequally distributed measurements from a system (see Fig. 3.18) and assuming fixed hyperparameters, the covariance matrix can be evaluated. The inefficiency of the algorithm becomes clear, since the large covariance matrix has to be inverted, though it implements many unimportant entries.

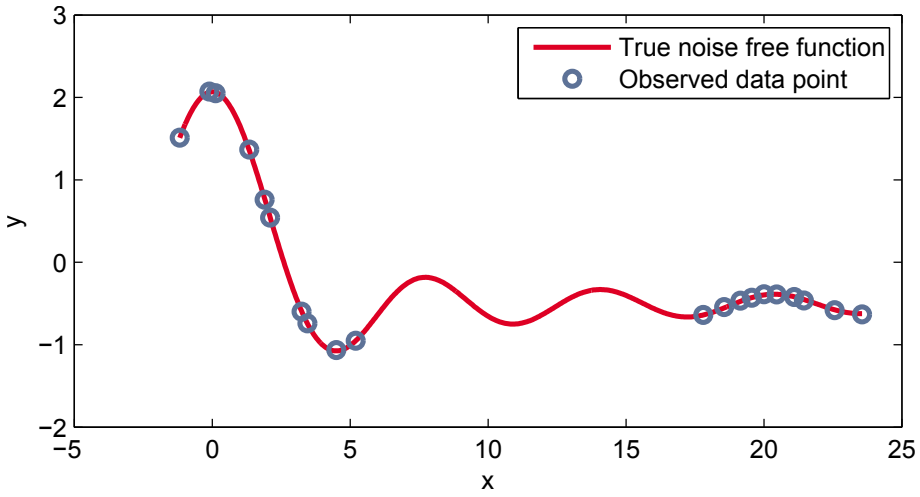


Figure 3.18: Measurements from the sinc-function (Tietze et al. 2014b)

By sorting the input space from lower to upper values, the covariance matrix is given as shown in Fig. 3.19. Here, for clarification, the covariance values below 0.01 are hidden. Since the distances between far-away input data points are large, the blockmatrices lying off the diagonal have almost no impact.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	x_{19}	x_{20}	x_{21}
x_1	1,14	1,04	1,00	0,69	0,54	0,49	0,24	0,21	0,11	0,09	0,05										
x_2	1,04	1,14	1,14	0,97	0,83	0,78	0,47	0,42	0,25	0,21	0,12										
x_3	1,00	1,14	1,14	1,02	0,89	0,84	0,53	0,47	0,29	0,25	0,15										
x_4	0,69	0,97	1,02	1,14	1,12	1,09	0,86	0,80	0,57	0,52	0,35										
x_5	0,54	0,83	0,89	1,12	1,14	1,14	0,99	0,94	0,72	0,67	0,48										
x_6	0,49	0,78	0,84	1,09	1,14	1,14	1,03	0,99	0,78	0,72	0,53					≈0					
x_7	0,24	0,47	0,53	0,86	0,99	1,03	1,14	1,14	1,05	1,01	0,84										
x_8	0,21	0,42	0,47	0,80	0,94	0,99	1,14	1,14	1,08	1,05	0,90										
x_9	0,11	0,25	0,29	0,57	0,72	0,78	1,05	1,08	1,14	1,14	1,07										
x_{10}	0,09	0,21	0,25	0,52	0,67	0,72	1,01	1,05	1,14	1,14	1,10										
x_{11}	0,05	0,12	0,15	0,35	0,48	0,53	0,84	0,90	1,07	1,10	1,14										
x_{12}												1,14	1,09	0,99	0,89	0,78	0,65	0,48	0,40	0,19	0,08
x_{13}												1,09	1,14	1,11	1,06	0,97	0,86	0,68	0,59	0,32	0,16
x_{14}												0,99	1,11	1,14	1,13	1,08	1,00	0,85	0,75	0,45	0,24
x_{15}												0,89	1,06	1,13	1,14	1,13	1,07	0,95	0,86	0,56	0,32
x_{16}						≈0						0,78	0,97	1,08	1,13	1,14	1,12	1,04	0,97	0,68	0,42
x_{17}												0,65	0,86	1,00	1,07	1,12	1,14	1,11	1,06	0,81	0,53
x_{18}												0,48	0,68	0,85	0,95	1,04	1,11	1,14	1,14	0,97	0,71
x_{19}												0,40	0,59	0,75	0,86	0,97	1,06	1,14	1,14	1,04	0,80
x_{20}												0,19	0,32	0,45	0,56	0,68	0,81	0,97	1,04	1,14	1,06
x_{21}												0,08	0,16	0,24	0,32	0,42	0,53	0,71	0,80	1,06	1,14

Figure 3.19: Evaluated covariance matrix for sorted input values (Tietze et al. 2014b)

- *Noise variations and adaptation of complexity*: The stationary GPs are not able to adapt functions of input-depending smoothness (Plagemann et al. 2008, Paciorek & Schervish 2004). The smoothness of the GP model is given by the

lengthscale hyperparameter w_j for each input j. The impact of the lengthscale was shown in Fig. 3.15. Small lengthscales make the model more flexible but also lead to overfitting. The opposite appears if the lengthscale is too high, then the model becomes inflexible and leads to underfitting. As can be seen, the lengthscale is constant after the training process and thus afterwards not dependent on the input values. However, next to the constant lengthscale hyperparameter, standard GPR also assumes constant noise level in every state-space position of the process. For real applications such as combustion engines, this assumption is certainly not realistic. Varying combinations of engine input parameter lead to different physical effects. If the dynamical system consists of such local effects, a full GPR model has to make a tradeoff between modelling the global process or the local effect. Thus the GPR model will lead to the effect of underfitting, since the global function dominates. In the figures, a function with a region of low noise (left side) and high noise (right side) is used. Furthermore, a local effect is added to the global function (middle). The GP model is not able to deal with the difference of noise in the data. Thus, the model becomes uncertain in the region of low noise, because of the noisy data on the right side.

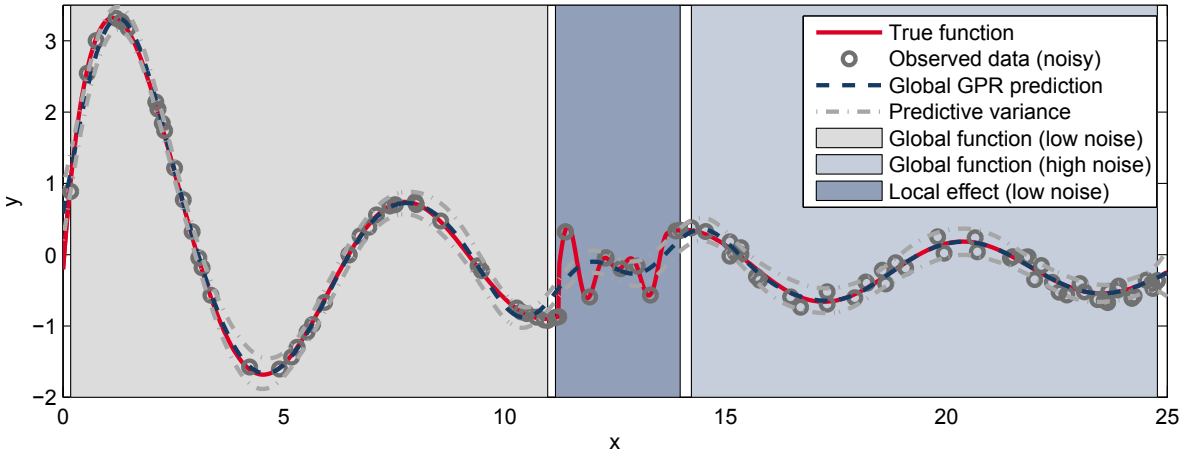


Figure 3.20: GPR model underfitting in local region (Tietze et al. 2014b)

- *Iterative modelling and model recalibration*: Generally, a trained GPR model consists of fixed hyperparameters which cannot be adapted or changed after the training process. If the GPR model predicts poorly for some test data, this region cannot be adapted by the model. The test data can be added to the old training data and the complete model can be retrained using the new data set. However, it is possible that the new data then leads to the effects of overfitting. Fig. 3.21

shows the effect of adding new data points in a local region to the global model. The local effects dominate and lead to a small lengthscale hyperparameter. Thus, the model overfits for the global function.

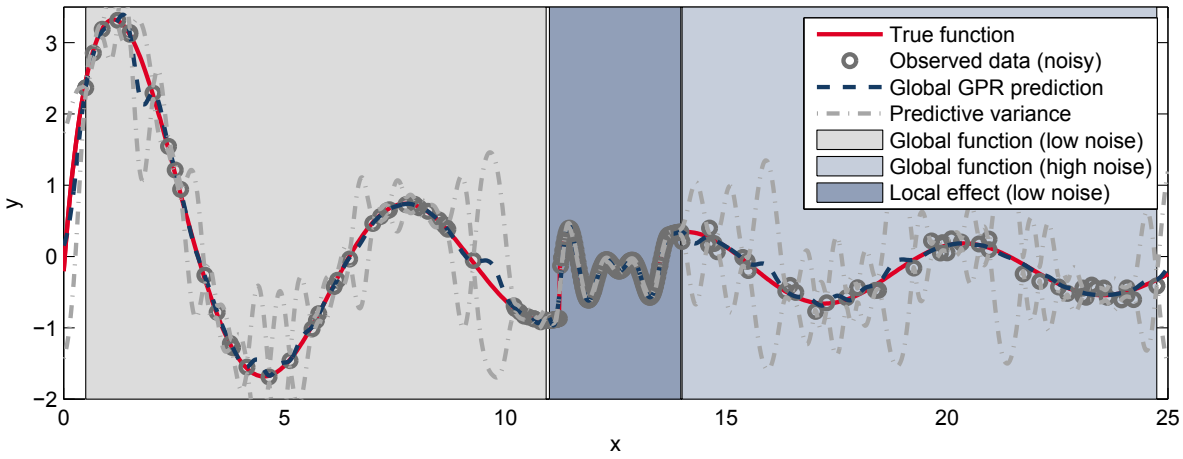


Figure 3.21: GPR model overfitting of a global function (Tietze et al. 2014b)

- *Adaption of changing dynamics*: Similar to the local effect, the dynamics of a system can change. The dynamical structure of the GPR model is shown in Fig. 3.22. When training the GPR model, the dynamical structure is also fixed due to the underlying regression problem. It is not possible to adapt the dynamic structure to a regression problem of changing system dynamics.

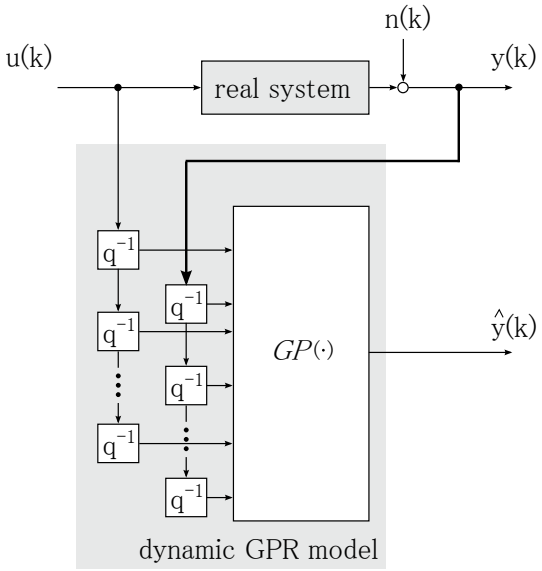


Figure 3.22: dynamical GPR model

The main drawbacks of standard GPR now become clear. In the following, the Local Gaussian Process Regression will be introduced in order to solve the aforementioned drawbacks of GPR for dynamic modelling and practical usage.

3.4.1 Local Gaussian Process Regression

Local Gaussian Process Regression was presented by Nguyen-Tuong et al. (2008) and Nguyen-Tuong et al. (2009) for real-time online model learning and control. The focus of the work in Nguyen-Tuong et al. (2009) was to speed up standard GPR for online applications. Therefore, the training data is partitioned into local regions and an individual GP model is trained for each one. The prediction for a query point is performed by a weighting estimation using nearby local models. The weighting of the models is defined by the distance of the query point to the local models. For M local models, the prediction for a mean value $\hat{y}$ is given as:

$$\hat{y} = \frac{\sum_{k=1}^{M} w_k \bar{y}_k}{\sum_{k=1}^{M} w_k}$$

where $\bar{y}_k$ is the local prediction and w_k is the measure metric given as

$$w_k = exp(-\frac{1}{2}(x - \boldsymbol{c}_k)^T \boldsymbol{W}(x - \boldsymbol{c}_k))$$

Here, $\boldsymbol{c}_k$ denotes the centre of the k-th local model and $\boldsymbol{W}$ a diagonal matrix represents the kernel width. Thus, each local prediction $\hat{y}$ is additionally weighted by the distance w_k between the corresponding centre $\boldsymbol{c}_k$ and the query point x (Nguyen-Tuong et al. 2009). However, weighting over the distance seems to be useful when the input space can be clustered distance-based. A modification of the algorithm was presented in Tietze et al. (2014b). In this paper, the key idea is to generate local GPR models and use the uncertainty of each model to weight the local model outputs to a mean output. In the next sections the concept, the algorithm and its advantages compared to standard GPR will be shown.

3.4.2 VW-LGPR - Concept

Given the example of the unevenly measured sinc-function in Fig. 3.18 and the entries of the covariance matrix of Fig. 3.19 the key idea of Local Gaussian Process Regression

is to partition the data-space into smaller local sub-spaces, i.e. the original data sets are divided into smaller subsets and for each subset GPR models are learned independently. By doing so, the impact of far-away points are neglected, while the nearby data points are grouped to local regions (Tietze et al. 2014b). Generally the process of grouping data in a intelligent way is called *clustering*. A cluster is defined as a group of data that are more similar to each other than data data belonging to other clusters (Nelles 2001). Thus the key step when using clustering methods is the definition of a similarity measure. For the LGPR concept the aim is to define clusters which contain the data of the local effects. Thus the best clustering method is the one that identifies similarity of the data points regarding the properties of the local effect. Here, the strategy can change from case to case. For instance if the local effect and its boundaries is well known then the clustering of the data can be done manually by the user. Since for the clustering only the input space is recognized it belongs to the *unsupervised learning*. This means that when using the clustering strategy either prior information is needed or that the distribution of the data helps to identify the local effect. Sometimes the input data give no information about the local effect. In this case using *supervised learning* can be a promising approach. However, for the example, Fig. 3.23 shows a distance-based clustering (e.g. k-means clustering, see Appendix A.2) of the input space. Clustering by distance is a good strategy for the LGPR algorithm since the evaluation of the covariance matrix of the GP model is based on distances. Thus the separation of data into clusters which are far away from each other will keep the maximum information of the global covariance matrix.

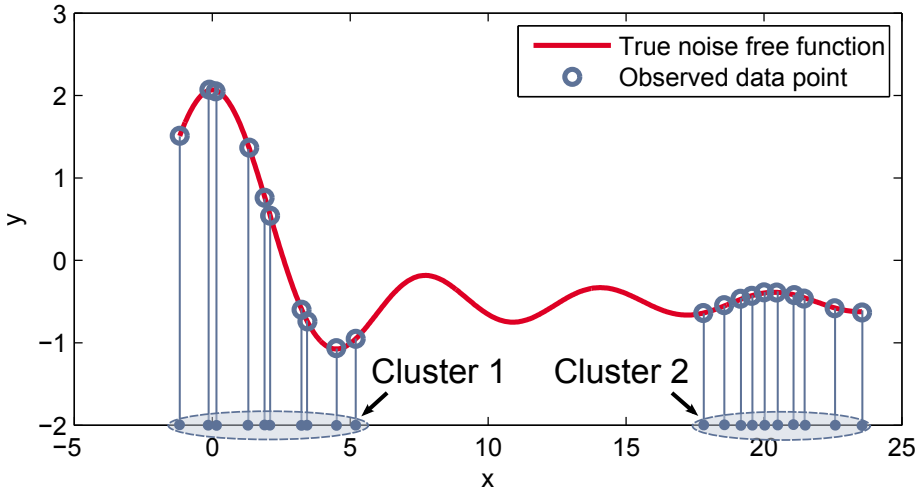


Figure 3.23: Clustering the input space (Tietze et al. 2014b)

For this example, we get two Local GP models, each with good prediction quality in the local training area, see Fig. 3.24.

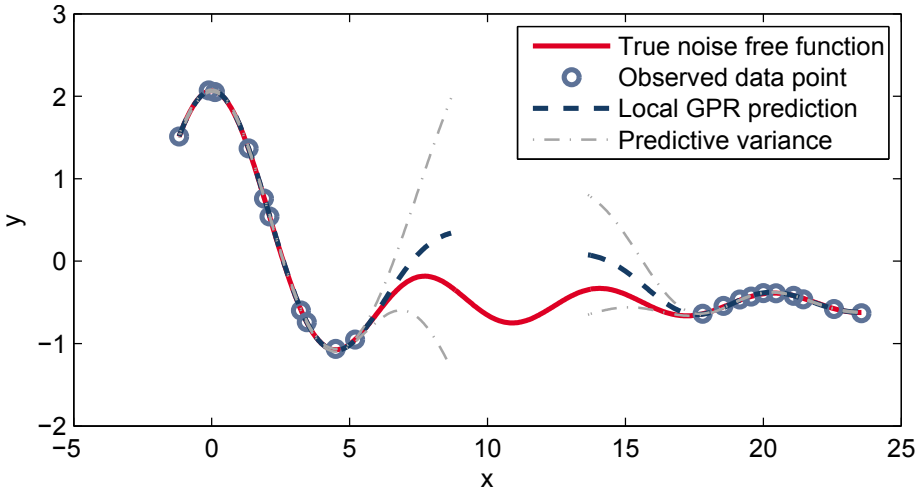


Figure 3.24: Prediction of two Local GP models (Tietze et al. 2014b)

The aim is to combine these two GP models to one coherent GP model. This can be done by weighting the predictive variances. As mentioned in chapter 3.3.3.3, the predictive variance can be interpreted as model uncertainty. The larger the variance, the more uncertain the data-based GPR model is about the prediction for the query point. The consistent combination of local GP models to achieve one single GPR model is given in the next sections. For this example the results are shown in Fig. 3.25.

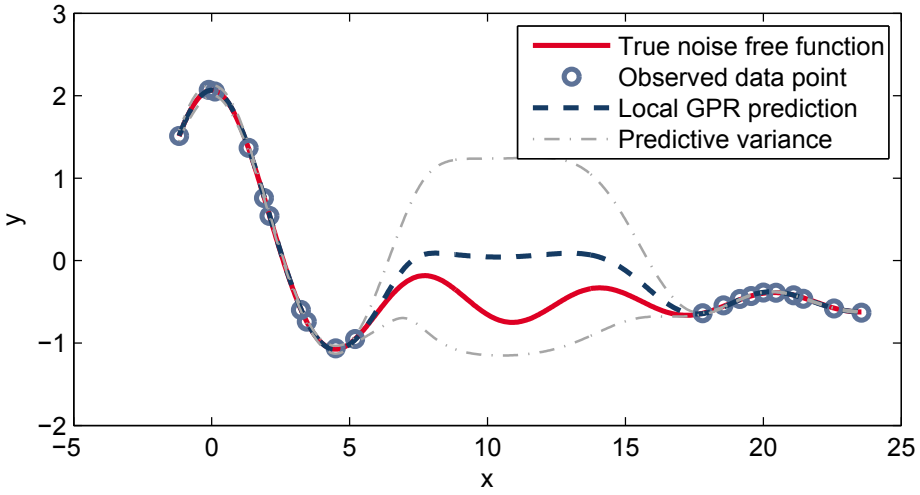
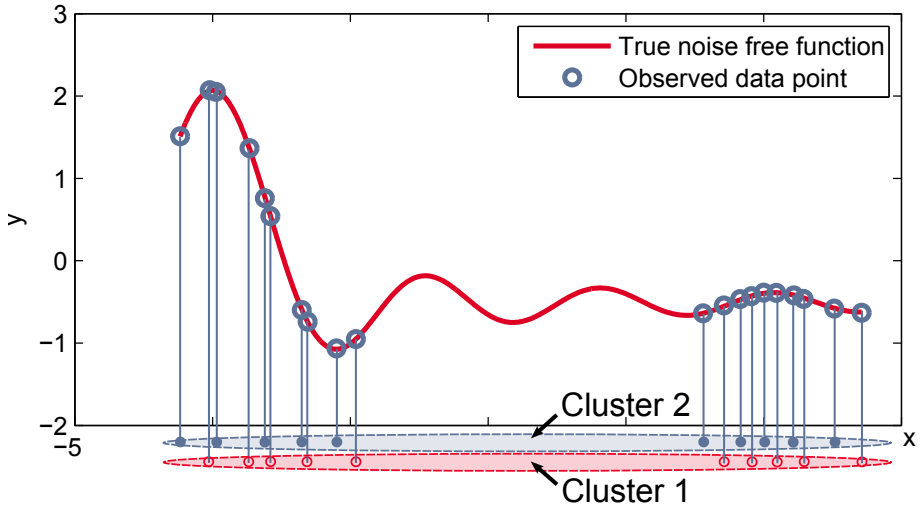


Figure 3.25: Coherent GPR model prediction by weighted variance (Tietze et al. 2014b)

3.4.3 VW-LGPR - Algorithm

Local GPR employs the standard GPR model described in chapter 3.3 as the basic technique. Here, the key idea is to partition the data space into smaller local subspaces, i.e. the original data sets are divided into smaller subsets, for each subset GPR models are learned independently (Nguyen-Tuong et al. 2009). Thus, the local GPR technique consists of 3 main steps:

- *Clustering and partitioning of the data*: As mentioned in the preceding chapter, the key idea of LGPR is to get a flexible modelling process by adding further local models. However, there will be a loss of information since the covariance function will not be evaluated for all input values. To minimise this information loss, a clustering of data is useful. Note that although good clustering leads to better models, LGPR is also robust towards suboptimal data clustering, see Fig. 3.26. Generally a good clustering means no overlapping of the different clusters and thus a minimum of information loss regarding the global GP model. For instance in the case of a suboptimal clustering the clusters contain nearly equivalent information and thus also have a strong overlap. The clusters and thus the GP models will not complement one another. With other words one cluster respectively one GP model will be useless and compared to the global GP model the half of information will be lost.



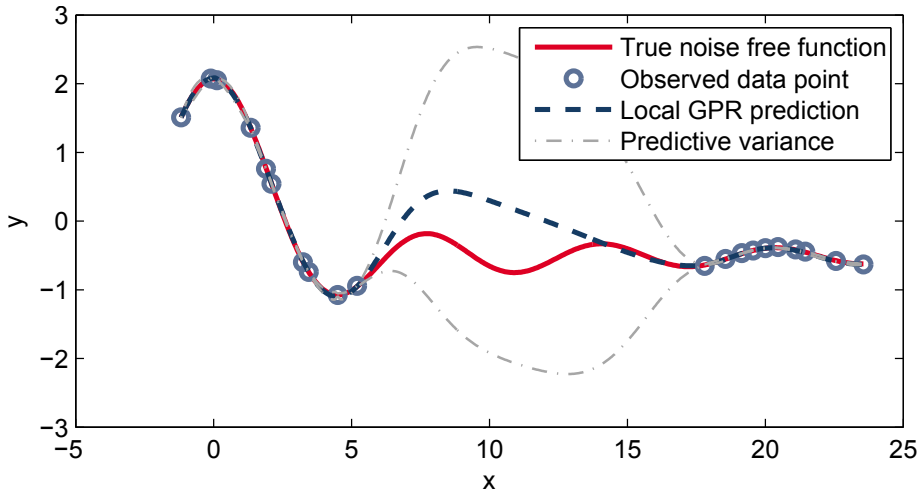


Figure 3.26: Prediction of Local GPs in case of suboptimal clustering (e.g. overlapping of local model regions) (Tietze et al. 2014b)

- *Learning and updating the local GPR models*: The second step is to train GPR models for each cluster, equivalent to chapter 3.3.3.3. Note that the training process will be more efficient, since in this example the dimension of the covariance matrix is reduced by half.

- *Combining the learned local models for a consistent prediction for a given query point*: Given a number of data clusters m, each cluster defines a training data set and, thus, we receive m mean functions $\bar{f}_*$ and predictive variance $cov(x_*)$ by using standard GPR. The mean of Local GPR can be computed by a weighted average $\hat{y}_{LGP}$ from means of the local models for a query point x_* (Nguyen-Tuong et al. 2009), i.e.

$$\boxed{\hat{y}_{LGPR} = \frac{\sum_{i=1}^{m} cov^i(\boldsymbol{x}_*)^{-1} \bar{f}_*^i(\boldsymbol{x}_*)}{\sum_{i=1}^{m} cov^i(\boldsymbol{x}_*)^{-1}} = \sum_{i=1}^{m} \pi_i(\boldsymbol{x}_*) \bar{f}_*^i(\boldsymbol{x}_*)} \tag{3.34}$$

with

$$\pi_i(\boldsymbol{x}_*) = \frac{cov^i(\boldsymbol{x}_*)^{-1}}{\sum_{i=1}^{m} cov^i(\boldsymbol{x}_*)^{-1}} \tag{3.35}$$

and

$$0 \leq \pi_i(\boldsymbol{x}_*) \geq 1 \qquad \sum_{i=1}^{m} \pi_i(\boldsymbol{x}_*) = 1 \tag{3.36}$$

Here, we employ the predictive variance $cov^i(\boldsymbol{x}_*)$ for weighting the corresponding

mean prediction $\bar{f}_*^i(\boldsymbol{x}_*)$ of each local GP model i. The basic structure of the weighted prediction is illustrated in Fig. 3.27.

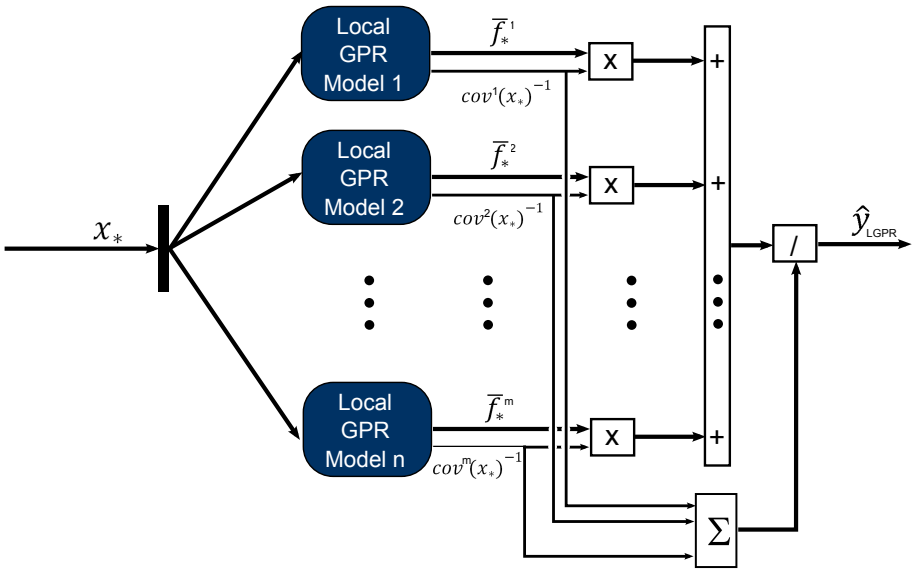


Figure 3.27: Basic structure of the weighted average mean prediction (Tietze et al. 2014b)

The corresponding predictive variance $\hat{\nu}_{LGPR}$ can be given as (see Appendix A.1.1):

$$\hat{\nu}_{LGPR} = \sum_{i=1}^{m} \pi_i(\boldsymbol{x}_*)(cov^i(\boldsymbol{x}_*) + \bar{f}_*^i(\boldsymbol{x}_*)^2) - \left(\sum_{i=1}^{m} \pi_i(\boldsymbol{x}_*)\bar{f}_*^i(\boldsymbol{x}_*)\right)^2 \tag{3.37}$$

Using these equations, mean prediction and predictive variance of VW-LGPR are given for a query point. Note that this approach is an extension of Nguyen-Tuong et al. (2009). In contrast to the work in Nguyen-Tuong et al. (2009), we employ the predictive variance for the weighted average $\hat{y}_{LGP}$ and, additionally, compute the variance $\hat{\nu}_{LGPR}$ for the corresponding weighted prediction. These results can be obtained when a Gaussian matching is performed on the mixed Gaussian densities resulting from local Gaussian density distributions (Tietze et al. 2014b).

Example:

Given the measurement from the sinc-function with a region of low noise (left), high noise (right) and a local effect (middle), the first step in order to generate a good model is to cluster the data adequately, see Fig. 3.28

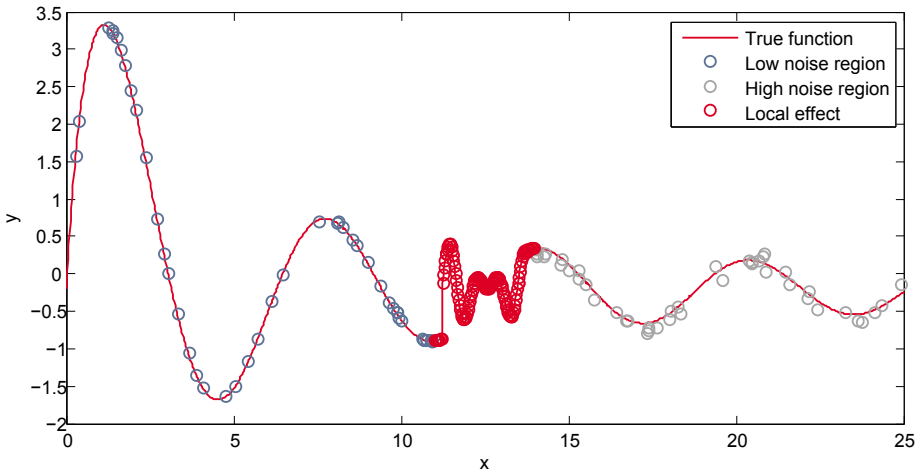


Figure 3.28: Clustered data

In the second step, for each cluster, a local GPR model is trained. In Fig. 3.29, two local GPR models learned appropriate noise levels.

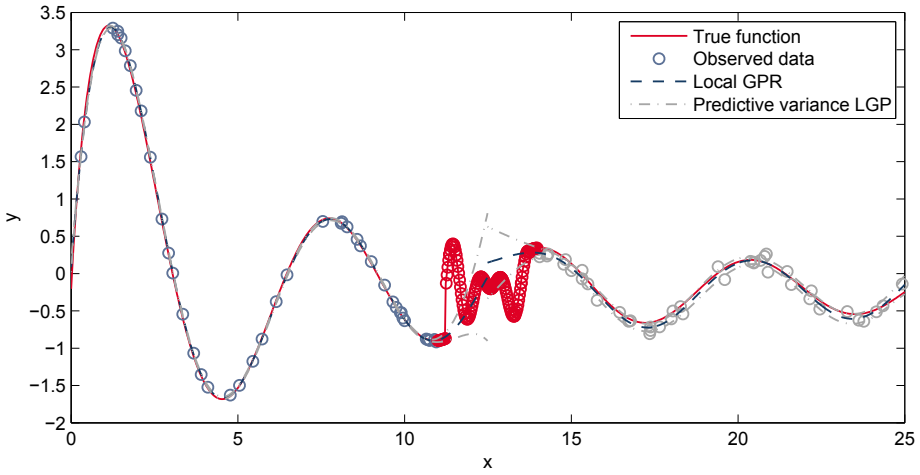


Figure 3.29: Local GPR in the low noise and the high noise region (Tietze et al. 2014b)

Also, the local effect can be exactly modelled by the local GPR model, see Fig. 3.30. It can also be seen that the GPR model for the local effect is much more flexible than both of the other ones.

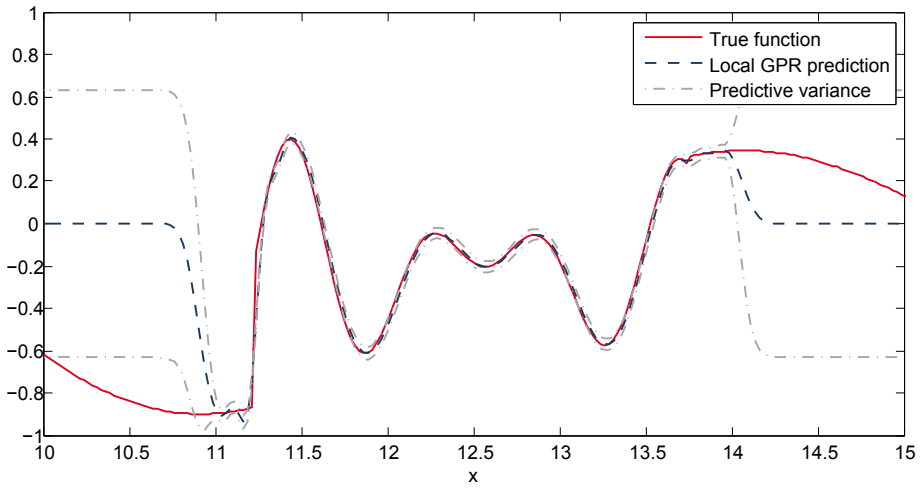


Figure 3.30: Local GPR at the local effect (Tietze et al. 2014b)

Using the variance weighted prediction of Eq. 3.34 and the variance approximation of Eq. 3.37 leads to the coherent VW-LGPR prediction as shown in Fig. 3.31.

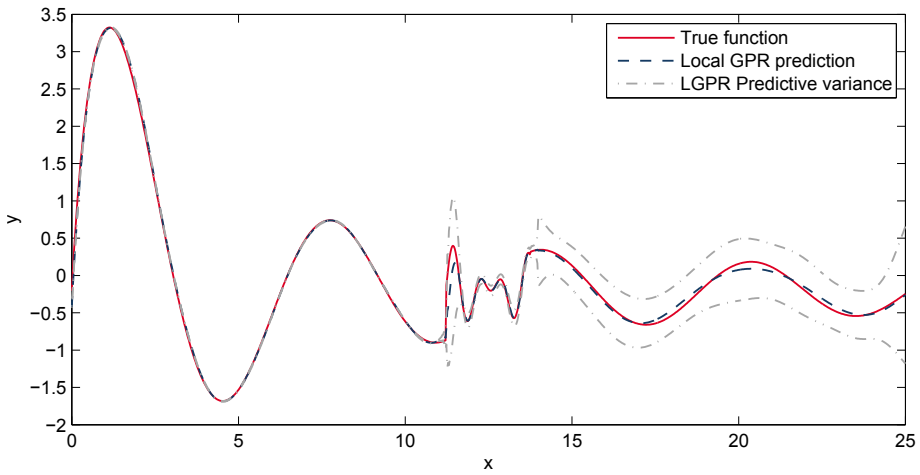


Figure 3.31: Coherent VW-LGPR prediction (Tietze et al. 2014b)

The validity of each model can be plotted by the normalised predictive variance of each local GPR model, as shown in Fig. 3.32. As can be seen the behaviour of the variance depends on the particular GP model. For the LGP1 and the LGP2 model the lengthscale is much larger than for the LGP3 model. This directly impacts the behaviour of the predictive variance. A flexible model like the LGP3 model (small lengthscale) becomes uncertain very fast while the uncertainty of a GP model with high lengthscale decreases slowly.

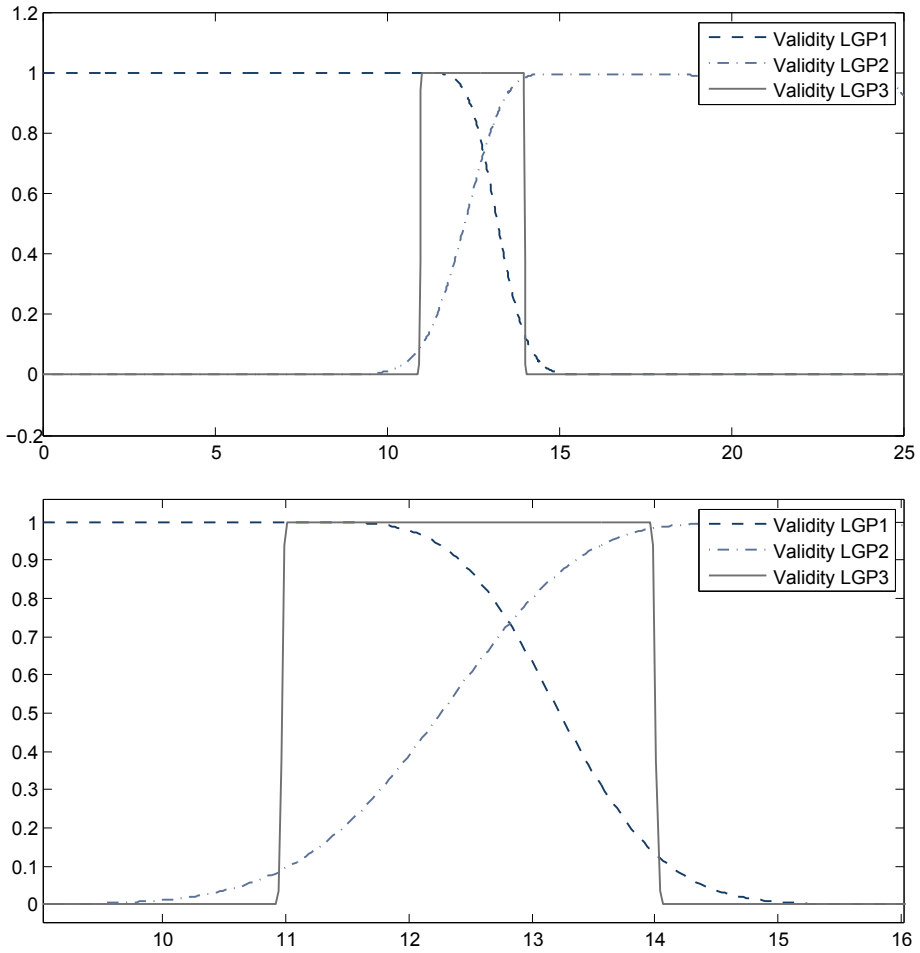


Figure 3.32: Validity of the VW-LGPR models

3.5 Summary of data-based modelling

The following table compares the approach of LGPR to the other algorithms of chapter 3.2. It can be seen that VW-LGPR combines the powerful regression performance of standard GPR with the advantages of LOLIMOT regarding local modelling. The disadvantage of VW-LGPR is related to it's performance. Thus for applications, where the requirements to speed and memory are high, the need of computing the variances of each local model is the drawback of VW-LGPR algorithm.

Properties	Polynomial	MLP	LOLIMOT	HILOMOT	GPR	VW-LGPR
Accuracy	-[1]	+[1]	0[1]	+[1]	++	++
Parameter estimation	++[1]	--[1]	++[1]	+[1]	--	-
Structure optimisation	-[1]	_[1]	+[1]	+[1]	++	++
Sensitivity to noise	+[1]	++[1]	++[1]	+[1]	++	++
Interpretation	0[1]	--[1]	++[1]	+[1]	++	++
Incorporation of prior knowledge	-[1]	--[1]	++[1]	+[1]	0	+
High dimensional mapping	--[1]	+[1]	-/0[1]	0/+	++	++
Large data sets	++	0	+	0	--	0
Training speed	+[1]	--[1]	++[1]	+[1]	-	+
Uneven data distribution	--	++	n.n.	n.n.	0	+
Expression of uncertainty	-	--	0	0	++	++
Weakening extrapolation	--	0	+	0	++	++
Iterative modelling	--	--	++	++	--	++
Model recalibration	--	--	++	++	--	++
Noise variations	--	--	+	+	--	+
Adaptation of local complexity	--	++	++	++	--	++
Adaptation of local changing dynamics	--	--	++	++	--	++
Simulation speed	0[1]	+[1]	+[1]	0[1]	-	--
Requirement of memory	++	+	0	+	-	--
Online adaptation	-[1]	--[1]	++[1]	++[1]	--	++
Effort for ECU implementation	++	+	+	0	-	--
Usability	0	-	++	++	++	+

[1] Nelles (2001)

4 Applications

In this chapter real-world applications for model-based ECU calibration will be shown. In order to provide the VW-LGPR features, a user-friendly graphical user interface (GUI) was developed in MATLAB (The MathWorks Inc.). The main GUI is shown in Fig. 4.1. The tool is able to load many measurements (see listed measurements in ①) and to set the measurement as training data or test data. It is therefore possible, for example, to combine different driven DoE measurements to achieve a set of training measurements. In ② all measurement values are shown. Here, it is necessary to check if all values are available in all measurements. The user can select the relevant values and transfer them to the input side (③) or to the output side (④).

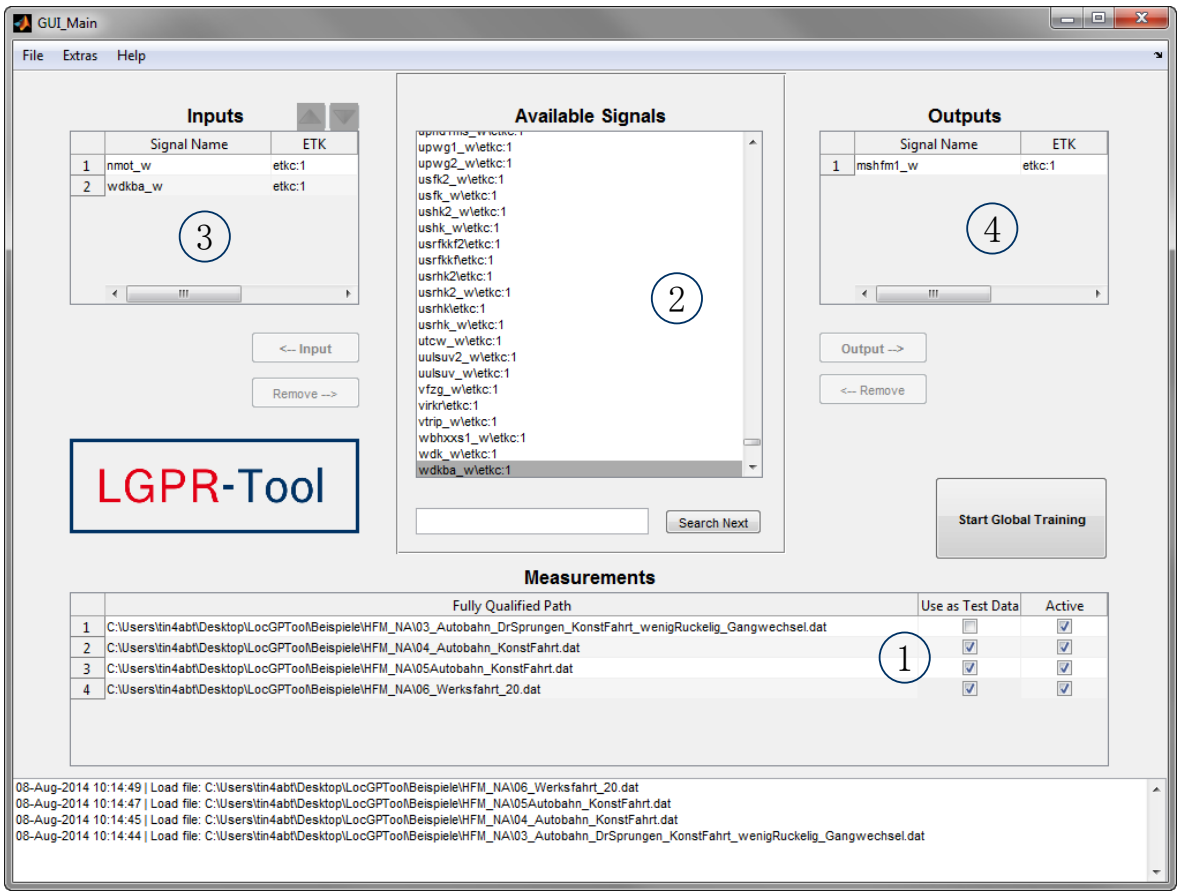


Figure 4.1: Main GUI of the LGPR tool

By pressing the *Start Button*, the NARX GUI allows the user to select the regressors which define the dynamical structure, see Fig. 4.2.

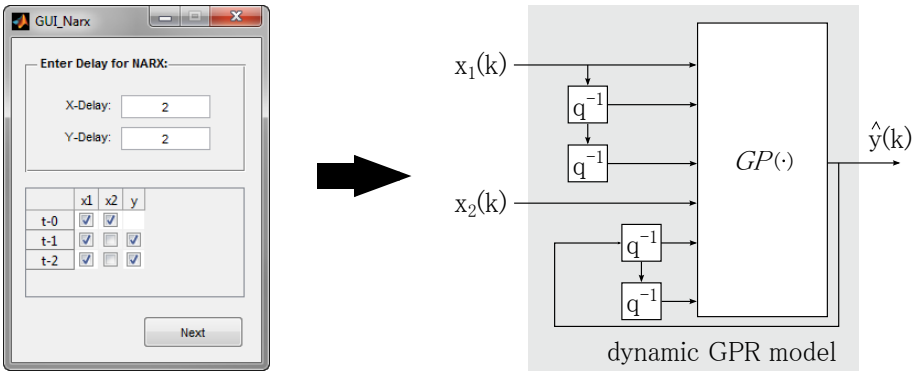


Figure 4.2: NARX GUI of the LGPR tool

In the next step the *Modelling GUI* is shown (see Fig. 4.3) The upper plots ① show

the input signals. The output is shown in ②. By selecting the tabs in ③ the user can select between the measurements which have been selected in the main GUI. In ④ the number of training points is shown. The pop-up menu in ⑤ provides different clustering methods, which can be selected by the user in order to cluster the training data into useful sample cluster. Furthermore, the methods can be used for downsampling. In this example, the 3026 sample data were downsampled (using the algorithm of section 2.6) and split into two clusters which should be approximately 250 sample points (see ⑤). The result of the clustering is shown in ⑥.

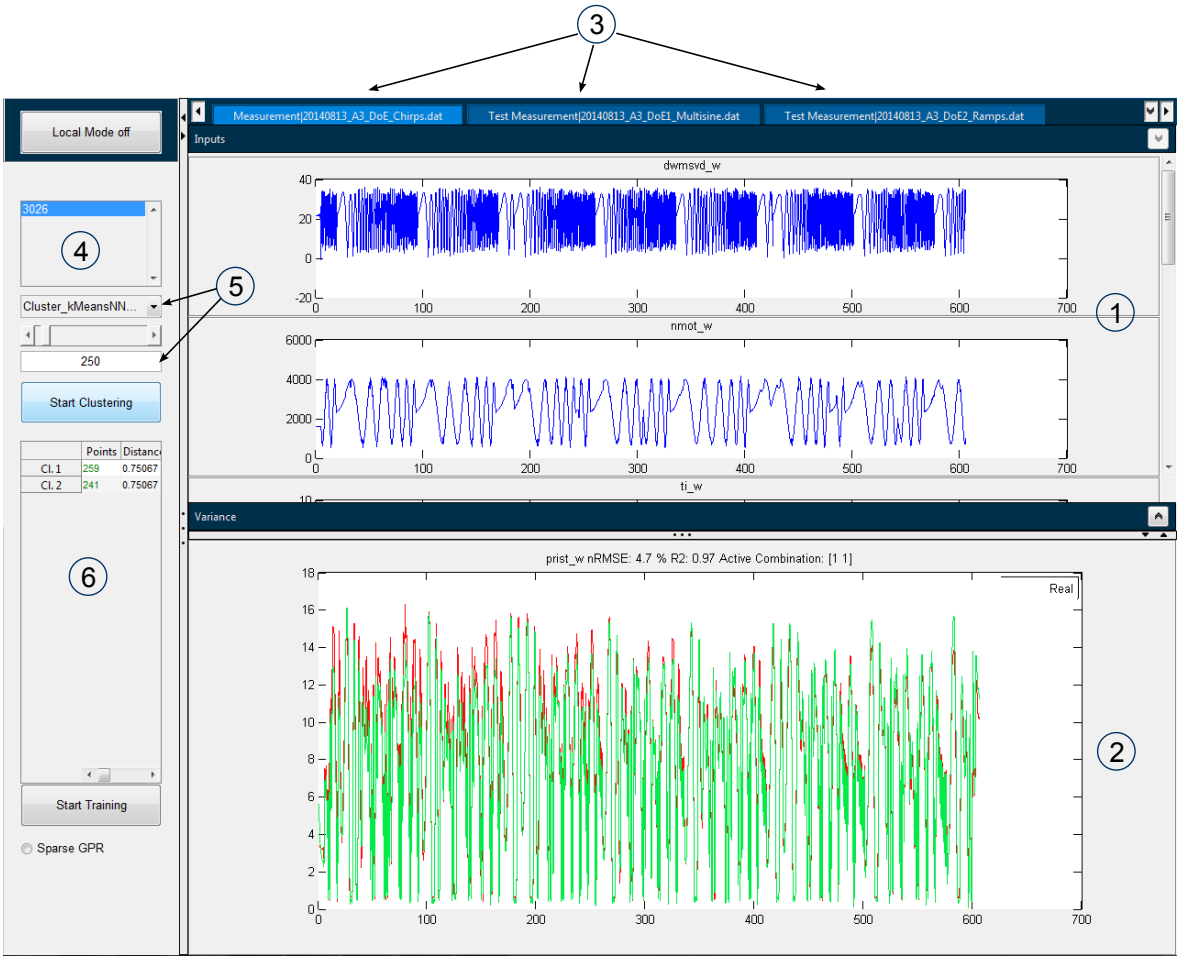


Figure 4.3: Model training GUI of the LGPR tool

The result of the training is shown in ②. Here, a R-squared value (R^2) of 0.97 indicates

a good training result. The R-squared value is a relative measure for the individual model errors on the validation data set and given as:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - y_{*i})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4.1}$$

where y_i are the measurement values, y_{*i} the model values and $\bar{y}$ the mean of the measurement values. However, in the next step, the test measurements have to be evaluated (see ③).

Figure 4.4: Model recalibration in the training GUI of the LGPR tool

The tool allows the predictive GP variance to be viewed (see Fig. 4.4) and marks those points which have a high predictive variance (see the inverse variance in ①) and thus indicates bad predictions (see ②). Now the user can choose if the marked points are to be used for a new local model or if these points shall be attached to the existing local models. Note that only the changed local models have to be retrained.

4.1 Close-PI Effect (stationary problem)

In diesel engine calibration a typical task is to optimise the engine noise due to the different engine states. The noise of a diesel engine is mainly influenced by the *air mass*, the *boost pressure*, the *pilot injection*,the *rail pressure*, the *start of control*, and the *delay between pilot injection and main injection*, see Fig. 4.5.

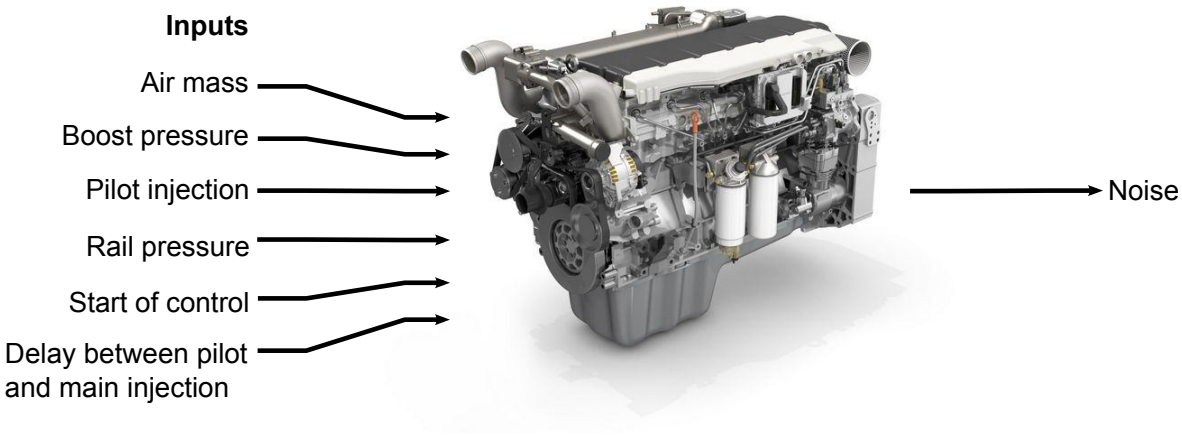


Figure 4.5: Inputs of the diesel engine due to engine noise

Especially the last input, the delay between pilot injection and main injection, has a strong nonlinear effect on the engine noise, which furthermore appears locally when the delay between the pilot and the main injection is small. Thus, the effect is called *Close-PI*. In order to visualise the effect, a global GPR model can be generated and the model output (noise) can be plotted over the input *delay between pilot injection and main injection*. The Fig. 4.6 is called *intersection plot*. The intersection gives information about the model output behaviour, when varying the chosen model input by keeping all other inputs constant. The dots are projections of the training data, thus they can be far away from the intersection. Training data samples which are near the intersection (Euclidean distance) are marked with circles in the plot. The GP model works with normalised data. Thus the data are normalised between -0.5 and 0.5. Additionally for the model output the mean is subtracted.

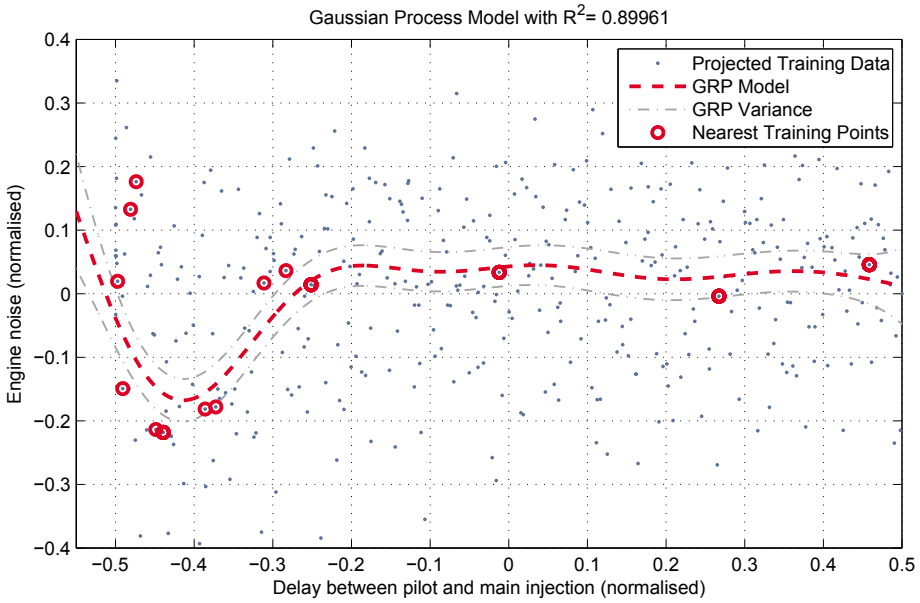


Figure 4.6: Visualisation of the Close-PI effect in an intersection plot

Fig. 4.6 shows the result of the GPR model when using a typical DoE measurement with 400 measurement points. It can be seen from the behaviour of the model that it does not fit the data well in the region of the Close-PI effect. In order to investigate the Close-PI, four different types of DoE measurements were made:

- DoE A: a Sobol design of 400 measurement points (see Fig. 4.7 a),
- DoE B: a design of 600 points with a higher density in the supposed Close-PI region (see Fig. 4.7 b),
- DoE C: a Sobol design of 600 points concentrated in the supposed Close-PI region (see Fig. 4.7 c),
- DoE D: and a measurement of 1591 points, in which centroids of a Sobol plan were used and only one dimension variates while all other inputs are kept constant (see Fig. 4.7 d).

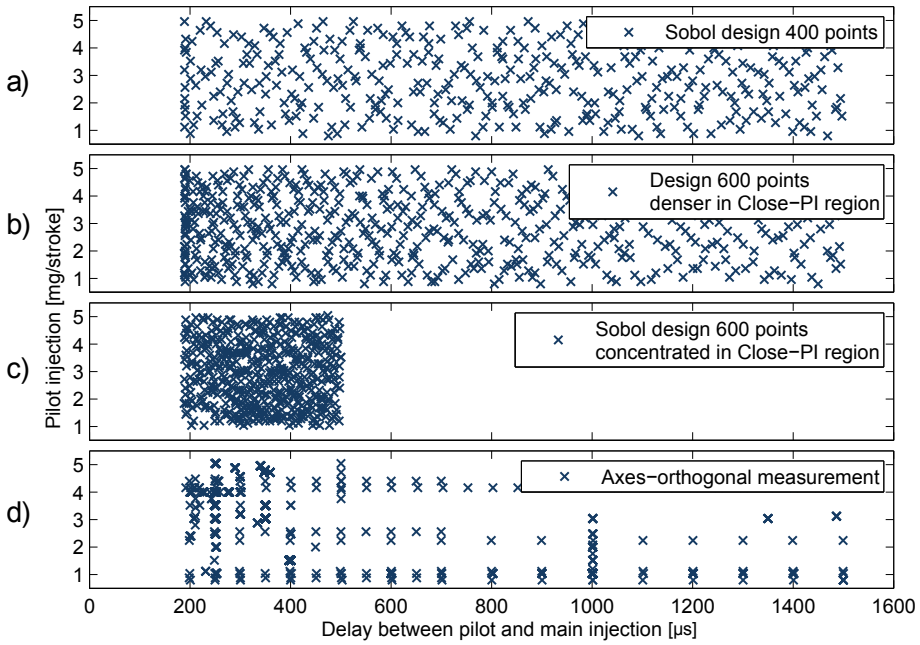


Figure 4.7: Different measurement designs in 2D

Using these measurements as training and test data produces the following results:

DoE plan	R^2 DoE A	R^2 DoE B	R^2 DoE AC	R^2 DoE BC	R^2 DoE D
DoE A (400P)	0.912 [1]	0.825	0.912[2]	0.730	0.443
DoE B (600P)	0.903	0.934 [1]	0.788	0.818[2]	0.489
DoE AC (1000P)	0.930[2]	0.869	0.880 [1]	0.867[2]	0.300
DoE BC (1200P)	0.910	0.948[2]	0.897[2]	0.913 [1]	0.519
DoE D (1591P)	0.000	0.000	0.305	0.273	0.998 [1]

[1] training result
[2] training data inside

It can be seen that none of the measurements are able to predict all the other ones with a high accuracy. Especially DoE D strongly differs from the other measurements. To get a better understanding of the underlying problem, all measurements are merged together and the HSFM algorithm of chapter 2.6 can be used to reduce the training data to an optimal distribution. In Fig. 4.8 the regression problem of the Close-PI effect is shown.

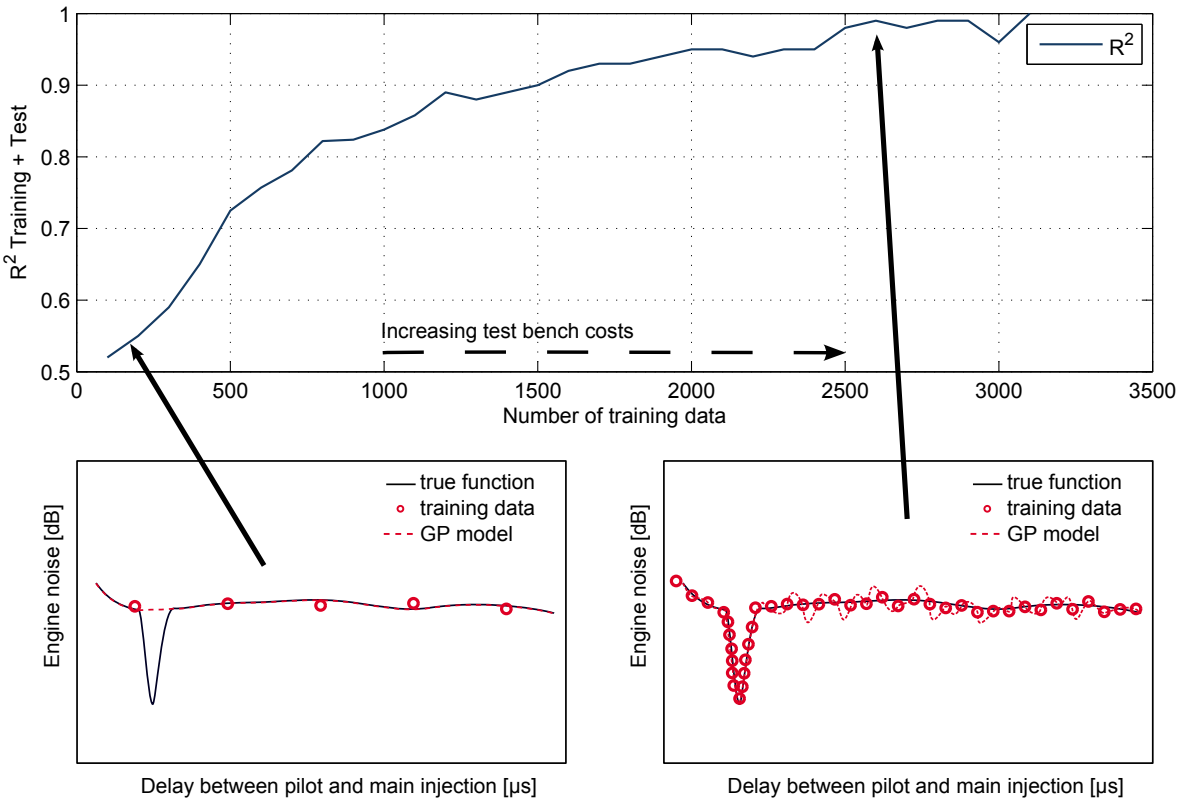


Figure 4.8: Close-PI regression problem

In the upper plot the R^2 is graphically displayed dependent on the number of training data. For example, using 500 training data, a R^2 of approximately 0.73 can be reached, when evaluating 2691 test data and the 500 training data. The regression problem of the Close-PI effect can be seen in the lower plots. Using a small number of training data, the global model behaviour can be modelled sufficiently but the local Close-PI effect is ignored completely (see left plot). Using a large number of training data the local effect can also be modelled sufficiently, but a small lengthscale hyperparameter was learned, so that the GPR model tends to overfit in the global region (see right plot). The disadvantage of GPR regarding uneven data distribution was mentioned in chapter 3. Also the number of measurements is not acceptable in the right plot regarding the test bench costs. In Fig. 4.9 the motivation for applying the LGPR algorithm for the regression problem of the Close-PI effect is depicted.

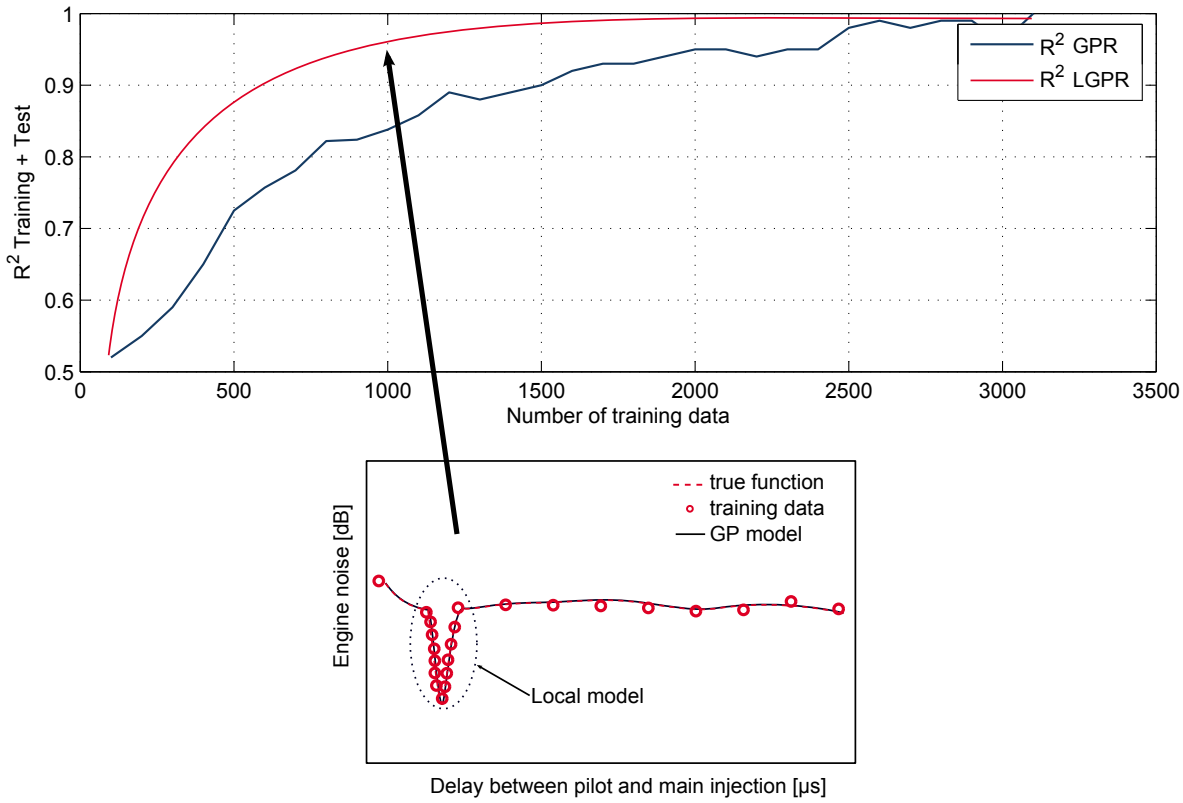


Figure 4.9: Motivation for applying LGPR

As pointed out in chapter 3.4.1, LGPR allows the combining of GPR models. Thus the motivation of using LGPR for this regression problem is being able to separate the local effect in order to generate a local model and combine it with the global one. The required measurement data for the global model is therefore small and the measurement can be concentrated on the local Close-PI effect. As shown in Fig. 4.9 the number of required training data can be significantly reduced.
However, the results of VW-LGPR are shown in Fig. 4.10 using different clustering strategies. The results of the k-means clustering indicates the influence of the chosen clusters. It becomes clear that the k-means clustering is not the best choice of clustering, since the aim here is to cluster exactly the local effect of the Close-PI region. A clustering method called *Close X far Y* searches for close distances in the input space where the output values (engine noise) strongly change. Also, manual centroids can be chosen and the input space data can be selected by their distance to these centroids. Using HILOMOT clustering (see chapter 3.2.3) also shows good results for the first 500 training points. Here, eight cluster were generated by the HILOMOT algorithm and used in combination with the HSFM downsampling in order to reduce the cluster to a

smaller number of training points. However, it can be seen that the clustering strategy is the key step in order to reach good results and that further investigations have to be done. Furthermore, in order to reduce the number of training data optimally, an optimisation strategy is needed to select the best training points. One approach could be to use the interpretable parameter of the GPR model (see chapter 3.2.4. Since the lengthscale hyperparameter indicates the flexibility of the model, the clustering could be done by starting with some data points outside the Close-PI region and train a GPR model. In the next step new training data can be chosen and a new GPR can be trained. Then the lengthscales of the model can be analysed. If the lengthscale hyperparameter and the variance do not change than the added points are not inside the Close-PI region. A significantly change of the hyperparameters indicates that the added points are inside the Close-PI region. However, this method seems to be time consuming and thus inefficient but since the Close-PI region is a local effect the strategy of adding points can be optimised.

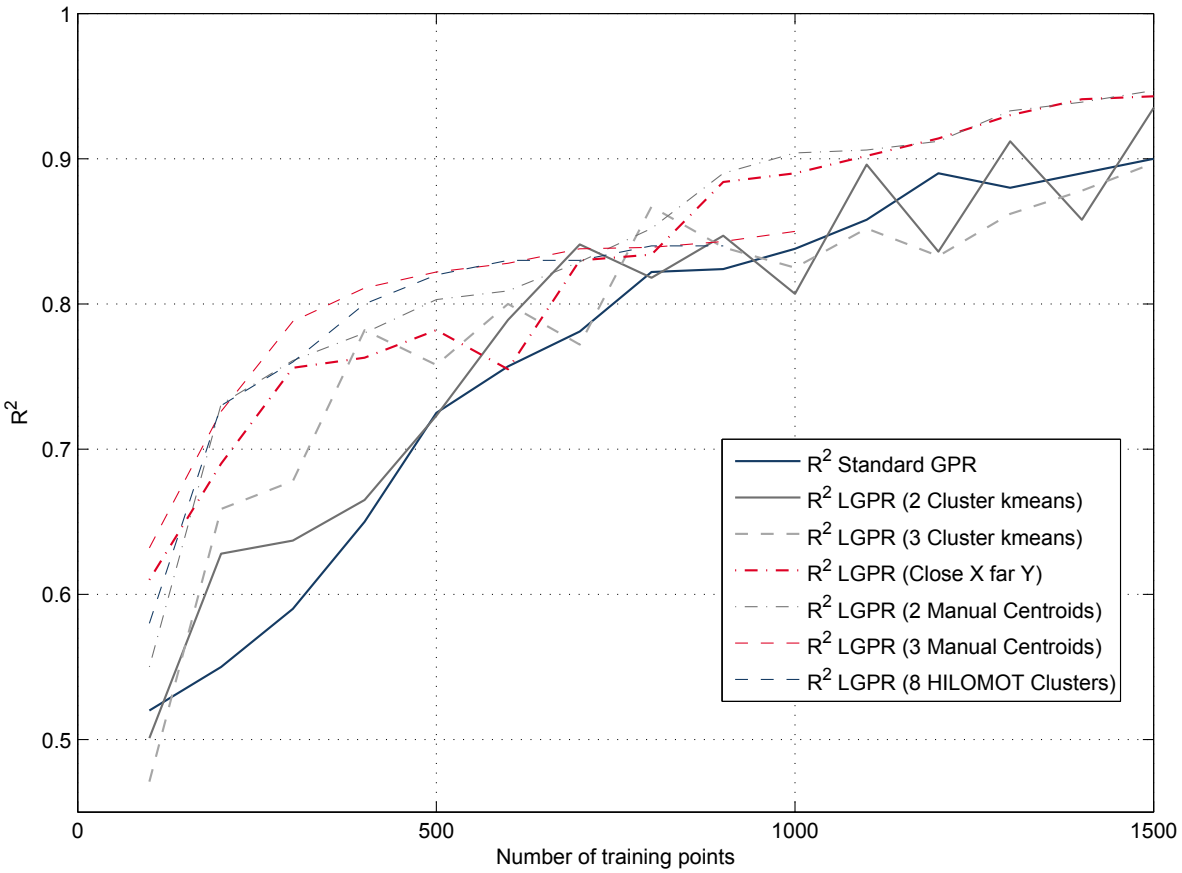


Figure 4.10: Results of LGPR using different clustering strategies

4.2 High Pressure Fuel Supply System (dynamical problem)

In this section the plant model of a control system will be identified. Typical control systems in a gasoline engine are:

- idle control,
- boost pressure control,
- lambda control,
- high pressure fuel supply control,
- camshaft control,
- knock control,
- throttle control.

The control systems differ in their level of difficulty regarding identification using data-based methods. As chapter 2 pointed out, the number of inputs should be small and they should allow external excitation. Furthermore, critical system inputs which could lead to system destruction complicate the identification process. Additionally, it simplifies the identification process if the system has a small nonlinearity and also a small system order and small dead times. Also, the benefit of the system model should be clarified beforehand, and thus the usage of the model for dynamical calibration should be high.
The High Pressure Fuel Supply control is well-suited for identification, since the number of inputs is small, it has weak nonlinear behaviour and a small system order. The High Pressure Fuel Supply (HPFS) of a modern gasoline engine is a demand-controlled system. A *high-pressure pump* (HDP) mechanically driven by the camshaft of the combustion engine delivers fuel into the fuel rail. The delivery rate is controlled by a *volume control valve* (MSV) inside the HDP. The PI-Controller of the ECU controls the MSV so that the fuel pressure inside the rail follows the set point depending on the engine operation. To keep the system protected from overload pressure, the HDP contains a pressure limitation valve. The main components of the high-pressure fuel supply system are shown in Fig. 4.11. The control process variable is the fuel pressure inside the rail, measured by a pressure sensor. From here, the electrical signal of the

sensor defines the feed-back signal for the PI-Controller. The command variable is the set point of the fuel pressure, which can be controlled by the actuating variable, here a *pulse width modulated* (PWM) control signal to the MSV. The calibration target is to determine the proportional and integral scaling coefficients to ensure optimal control of the fuel pressure inside the rail. The input of the MSV is given by the fuel low pressure, supplied by the low pressure fuel pump. The fuel flow through the MSV and the low pressure fuel pump is dependent on the battery voltage. In addition, the rail pressure is affected by the rail temperature and thus also by the ambient temperature. Given these inputs, the general overview given in chapter 2.2 can be used to get the types of inputs for the HPFS (see Fig. 4.12).

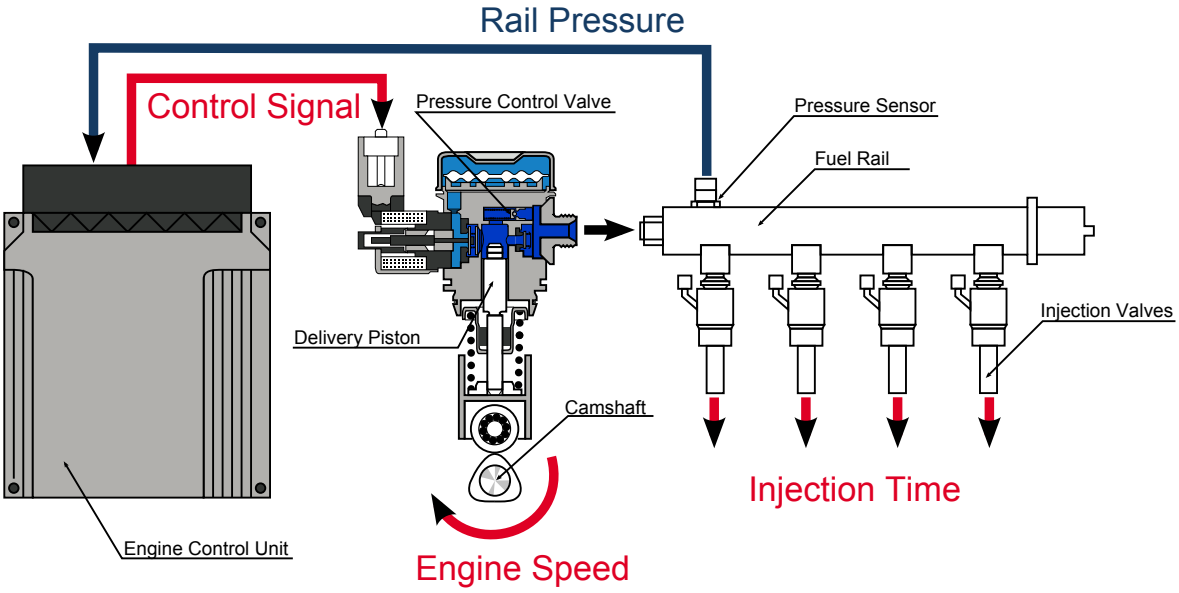


Figure 4.11: High Pressure Fuel Supply System (Tietze et al. 2014a)

Here the control loop of the HPFS control is open. The input MSV control is of input type 1, see chapter 2.2. Since the actuator (opening time of the MSV) is not measurable without adding additional sensors, and the resulting value (fuel mass) is also not measurable, the MSV control becomes the input and thus the MSV and high pressure pump become part of the dynamical system.

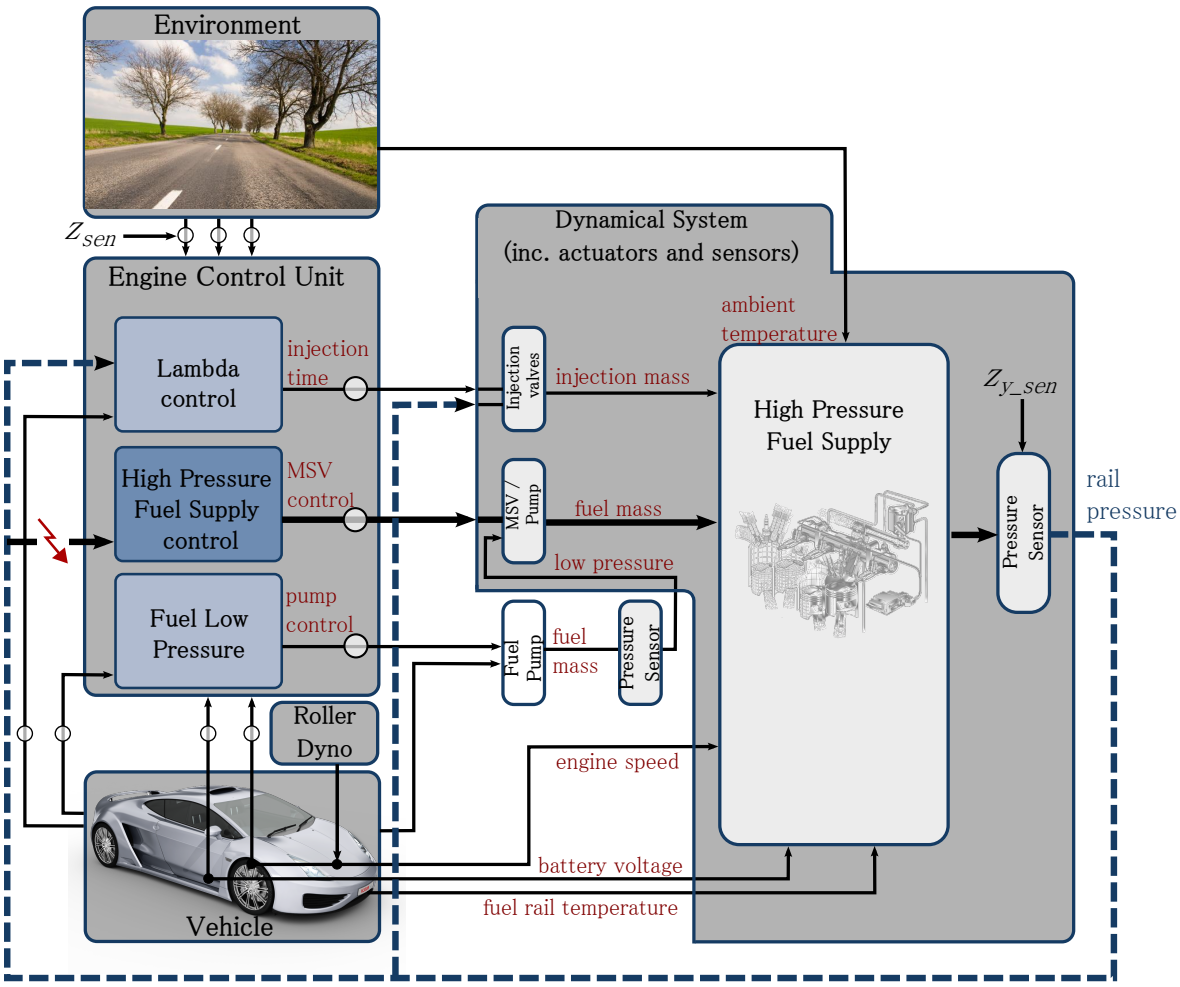


Figure 4.12: Types of inputs for identification of HDR plant (Tietze et al. 2014a)

As discussed in chapter 2.2, it is advisable to keep the model as simple as possible regarding input space dimension and the usage of the model. Thus, the importance of the different physical inputs should be evaluated. For the HPFS control plant model, the ambient temperature (type 6 input), the fuel low pressure (type 2 input), the battery voltage (type 4 input) and the fuel rail temperature (type 5 input) will be treated as disturbances. This can be done since the battery voltage and the fuel low pressure are almost constant in normal engine operation. The temperature of the rail is important, when the fuel flow through the rail is small, e.g., in phases of injection cut off. Aside from this special engine operation, the main inputs of the system are the MSV control signal (type 1 input), the engine speed (type 4 input), and the injection time (type 3 input), see Fig. 4.13. The injection time is a critical input, since it affects

the lambda of the engine combustion and thus the temperature of the exhaust. For this study, the injection time was not excited to preserve the system from destruction.

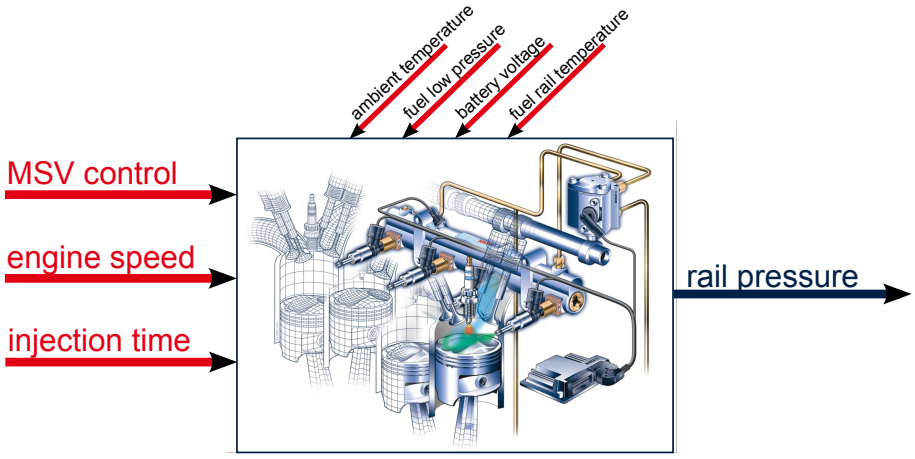


Figure 4.13: Simplified overview of the high pressure fuel supply control system (Tietze et al. 2014a)

The nonlinear behaviour of the system can be visualised by measuring step responses (see Fig. 4.14 a) and plotting the stationary values. The nonlinear function is approximated by a second-order polynomial (see Fig. 4.14 b). In Fig. 4.14 c the dynamical system response of the system is illustrated. Further nonlinearity tests can be found in Haber (1985).

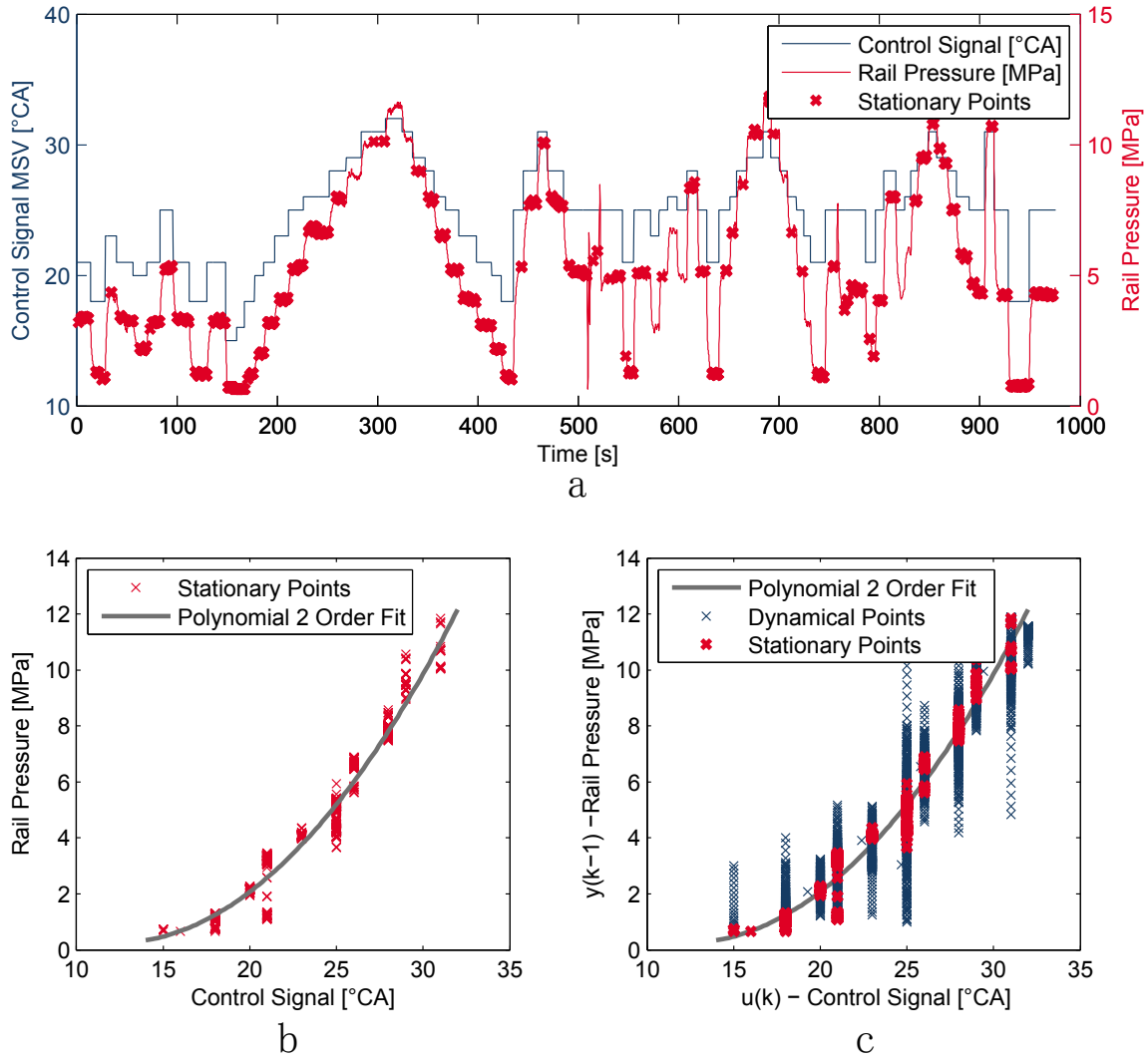


Figure 4.14: Nonlinearity test of the HPFS system: a) step response measurements, b) stationary measurements and polynomial fit and c) dynamical system response

4.2.1 Dynamical DoE for HPFS identification

As chapter 2 pointed out, the crucial step in dynamical DoE is to identify the permissible dynamical system boundaries. In the first step a stationary convex hull is identified and afterwards appropriate excitation signals can be scaled into the convex hull (see section 2.5.1). For the case of the HPFS system, the permissible stationary boundary is below the overload pressure of more than 15 MPa. The pre-identification of the convex hull can be done by using the HPFS controller [1] and setting the set point of pressure up to 15 MPa. Afterwards, the stationary safe boundary in engine

[1] Here we assume that the initially tuned PI controller can at least control stationary set points.

idle can be estimated by holding the engine speed at constant operating points and waiting until the controller sets the permissible control signal for the high pressure pump. Fig. 4.15 shows the result of the pre-identification measurement.

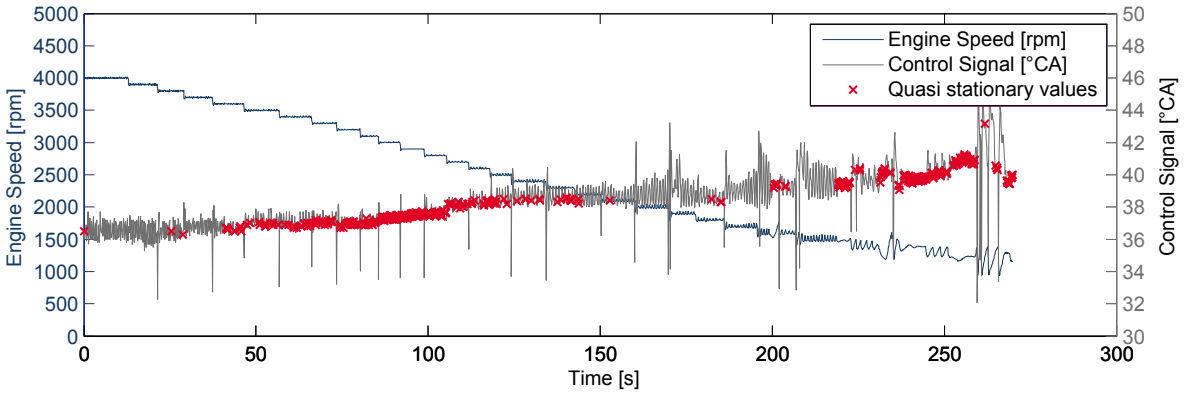


Figure 4.15: HDP input hull of control signal and engine speed in engine idle (Tietze et al. 2014a)

Given the stationary driveable measurements a convex hull can be computed (see section 2.5.1). Fig. 4.16 shows a scaled chirp DoE of engine speed and control signal. The quasi-stationary values are given by the pre-identification measurement, see Fig. 4.15.

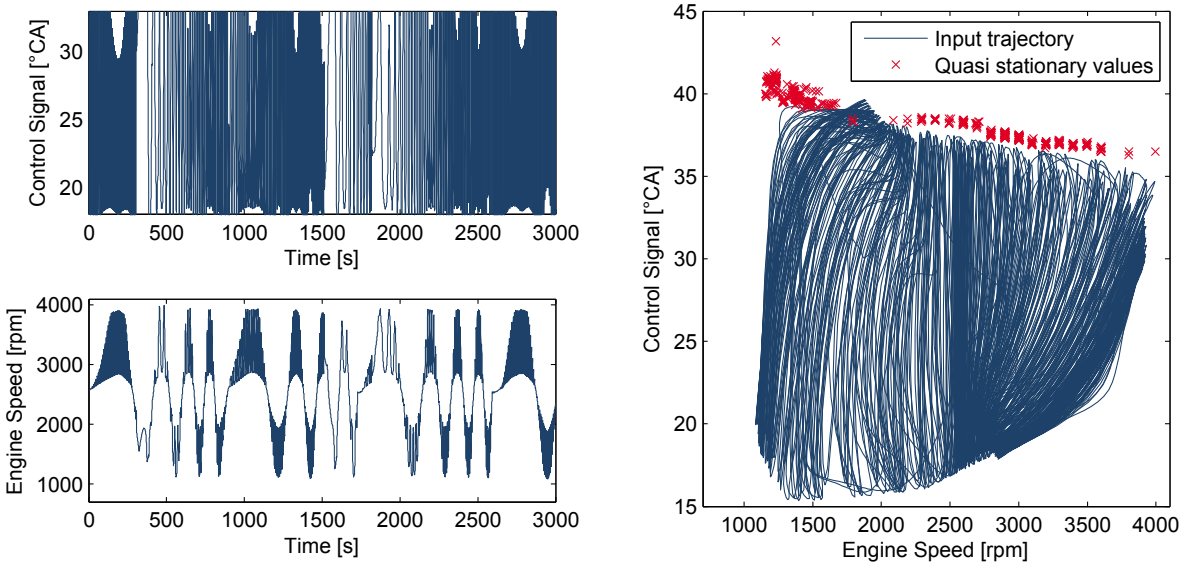


Figure 4.16: Chirp DoE scaled into the convex input space hull (Tietze et al. 2014a)

In Fig. 4.17 the different types of signals (APRBS, ramps, chirps and multisine) from

chapter 2.4 are scaled into the convex hull. The scaling method was presented in chapter 2 in section 2.5.1.

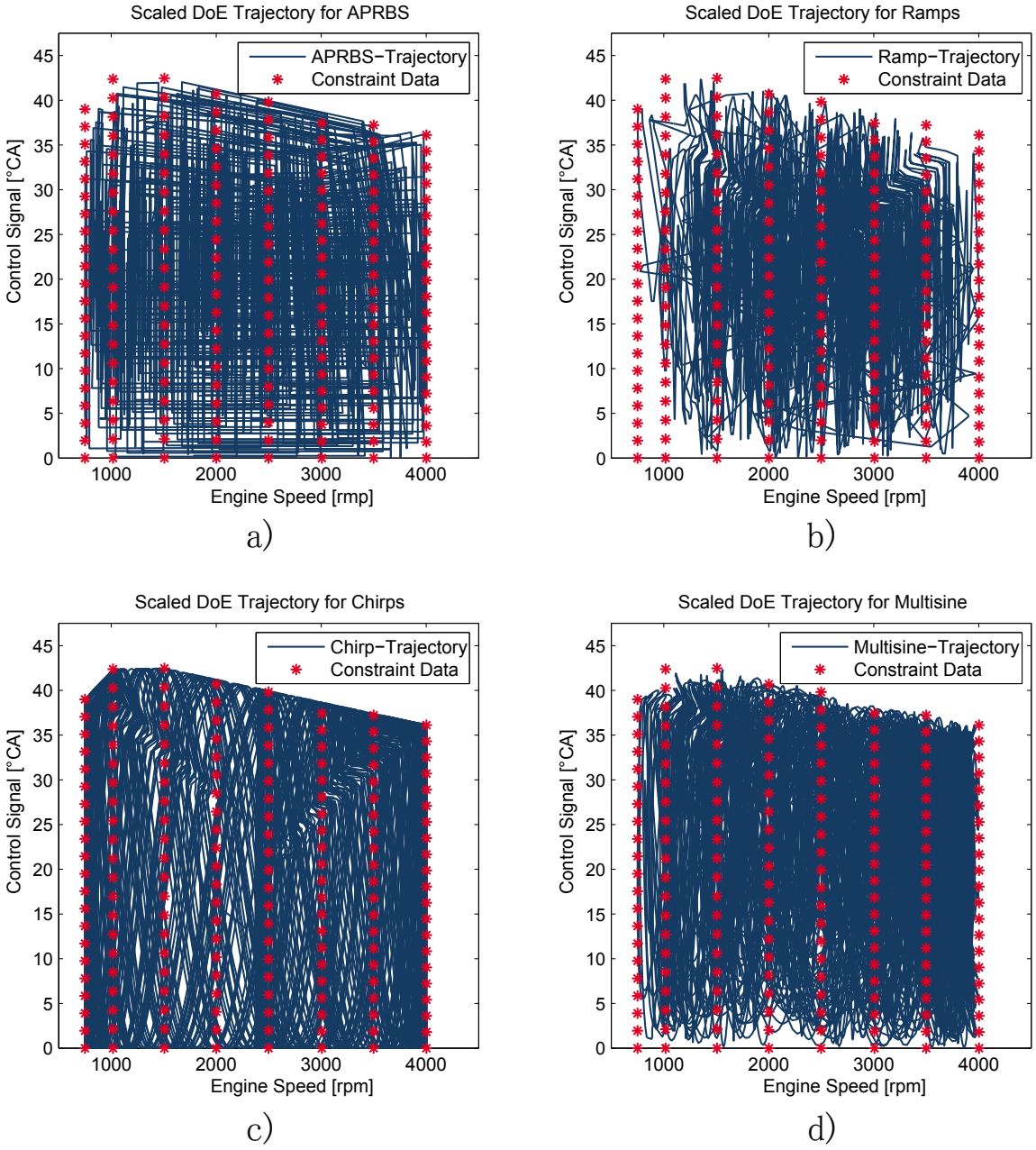


Figure 4.17: Scaled excitation signals: a) APRBS, b) ramp, c) chirp and d) multisine

In order to analyse the different signals before exciting the real system with them, the dynamic input space can be analysed, for instance, by plotting the gradients of the inputs and evaluating the space coverage (see Fig. 4.18).

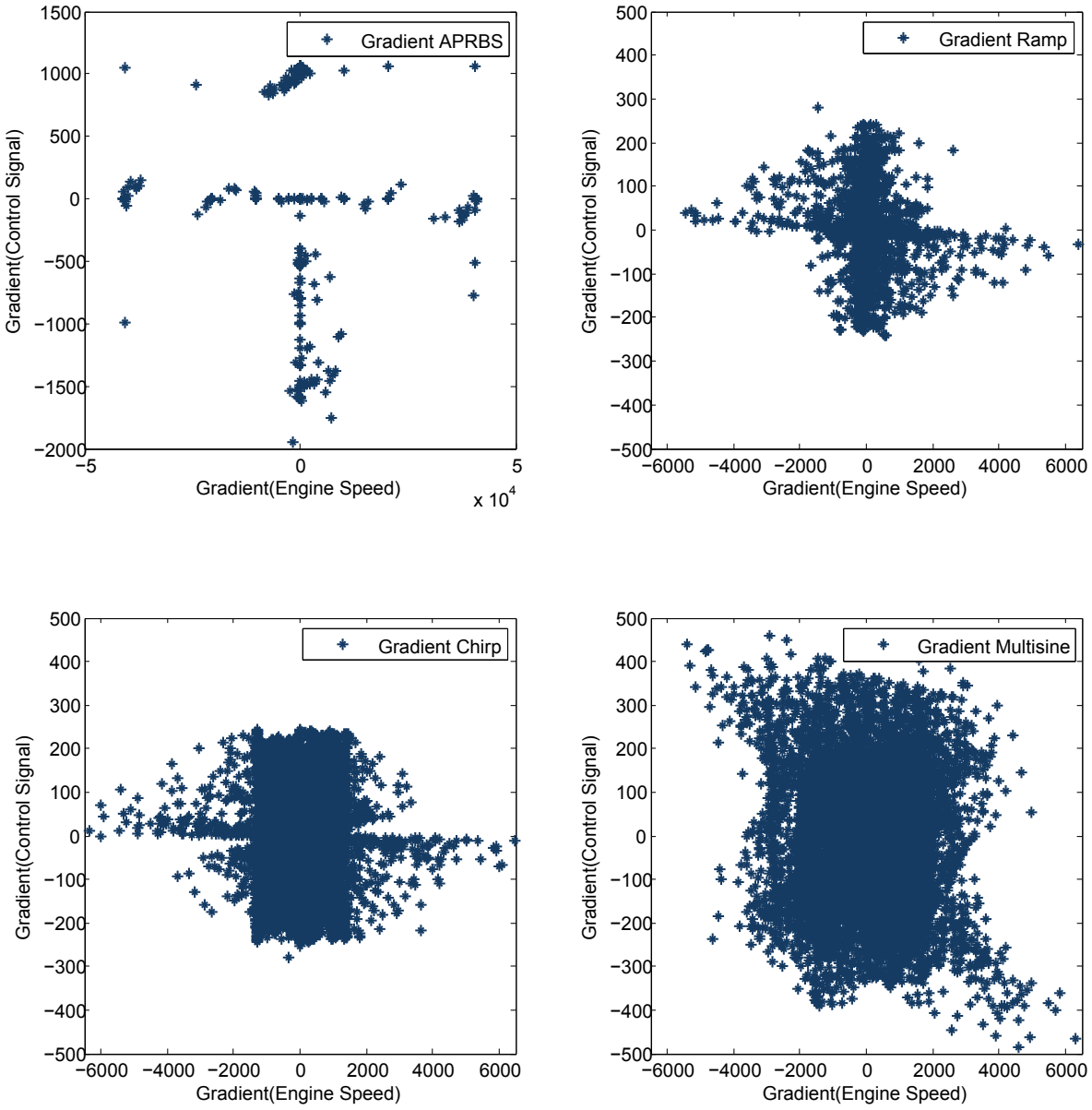


Figure 4.18: Gradients of excitation signals: a) APRBS, b) ramp, c) chirp and d) multisine

Regarding the coverages of the input spaces (Fig. 4.18 and Fig. 4.17) the chirp and the multisine show good properties. However, the distribution of the signals is missing so the HSFM value should give more information about the signals. Since the HSFM is a relative measure the change of the HSFM value can be interesting, see Fig. 4.19. The plot shows the space filling property when using a particular number of samples per dimension. The APRBS signal is generated by a Sobol space filling design. Thus it shows a high relative HSFM value for a small number of samples. But since the APRBS signal consists of long hold phases the HSFM decreases with the number of

samples per dimension. The ramp signal shows the best relative HSFM value since it is also generated by a space filling design.

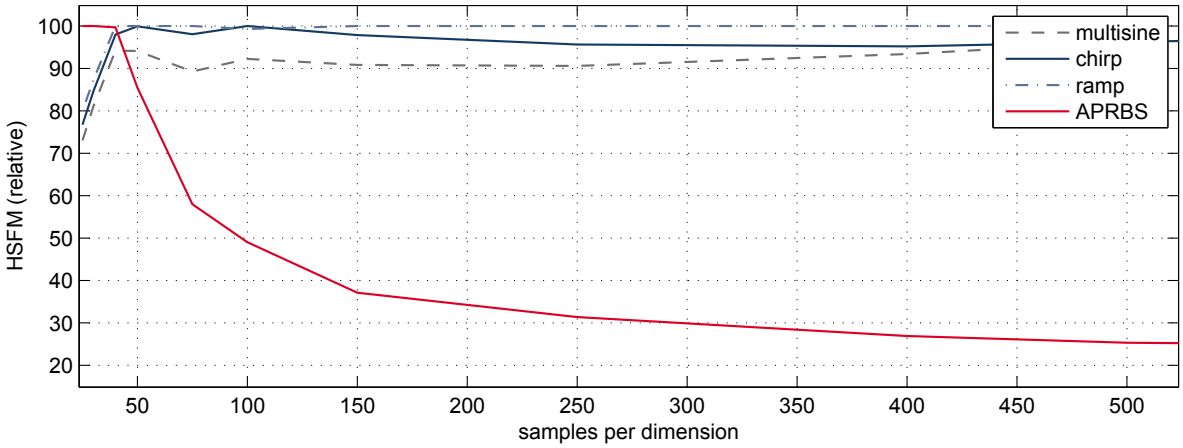


Figure 4.19: HSFM relative for chirp, multisine, ramp and APRBS DoE

However, the regression model also uses the delayed input values and furthermore the delayed model output. Thus the HSFM will be again evaluated after the DoE measurements in the next section.

4.2.2 Automated offline DoE

Given the dynamical DoEs from the preceding section the measurement must be automated in order to follow the given trajectories. The engine speed can be controlled by the engine speed limiter. The control signal can be set by a given software calibration interface. To transfer the DoE values to the ECU the standard calibration software INCA (ETAS GmbH) can be combined with MATLAB. Since the communication of the standard API is too slow (approximately 3 Hz), ETAS GmbH provides a tool for automation called INCA-FLOW and a prototyping and interface module called ES910. This configuration allows a very fast communication with the ECU (here a sample time of 0.02 seconds was chosen). As can be seen from Fig. 4.20 the engine speed limiter can control the given DoE plan precisely.

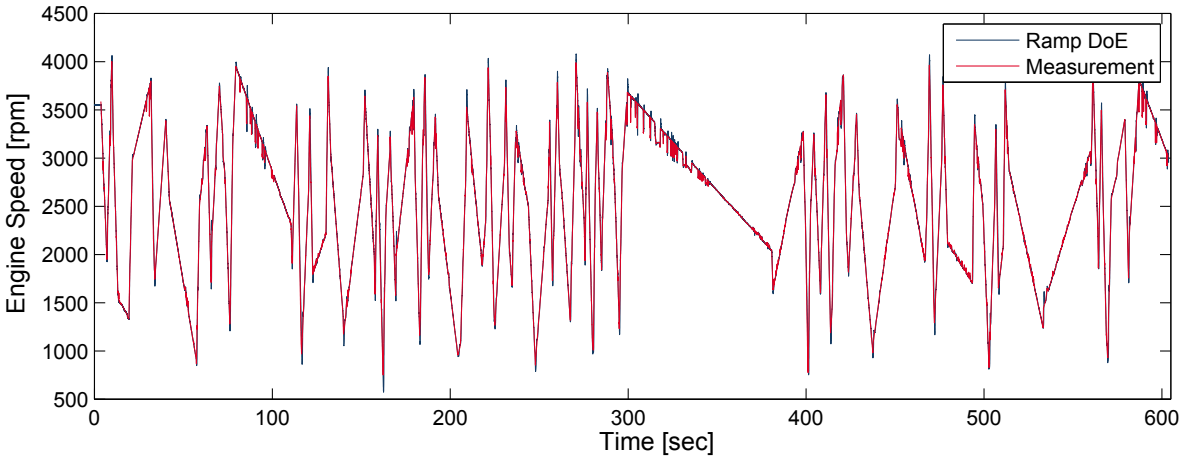


Figure 4.20: Measurement of a ramp DoE using the engine speed limiter

The APRBS DoE could not be measured, since the ECU is monitored and identifies the high engine speed gradients as an insecure engine state. The step excitation has already been mentioned as a major disadvantage of APRBS, see chapter 2.4.1. However, the ramp, the chirp and the multisine excitation can be measured. Fig. 4.21 shows the measurement of a chirp excitation. In the two upper plots the scaled chirps are shown. The lower plots show the system response and it can be seen that the pressure limit of 15 MPA is not violated.

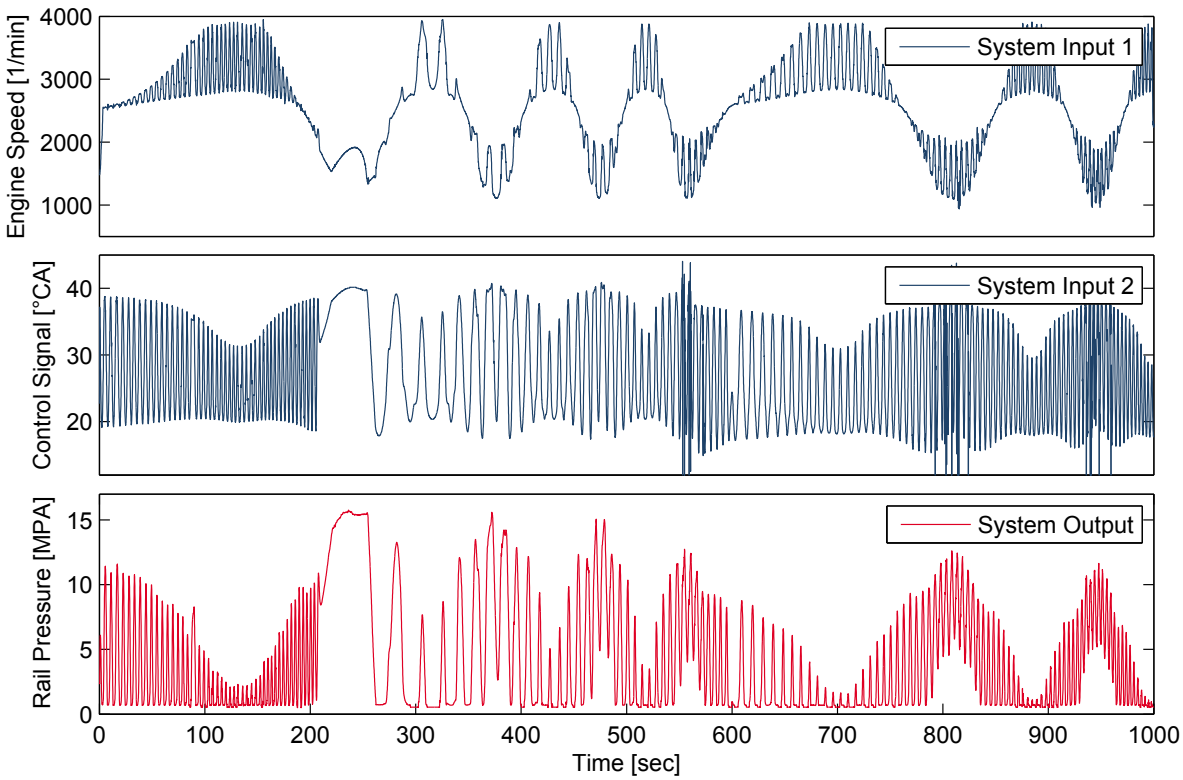


Figure 4.21: Automated measurement of a shifted Chirp DoE (Tietze et al. 2014a)

The measurements can be used to train GPR models, either for a large training set of approximately 30.0000 training points using sparse methods as implemented in ETAS ASCMO, or by using the HSFM downsampling and standard GPR. For all models the input and the output is delayed in order to achieve the dynamical structure. Thus a seven dimensional regression problem appears. The regressors are chosen by using a brute-force strategy. This strategy of trial and error only works if a suggestion about the system order exists. In this application a static model shows good results thus it indicates that a small system order can be assumed. By using the relative HSFM for variations of the samples per dimension for the model inputs and the output again gives information about the distribution of the data due to the space filling property, see Fig. 4.22.

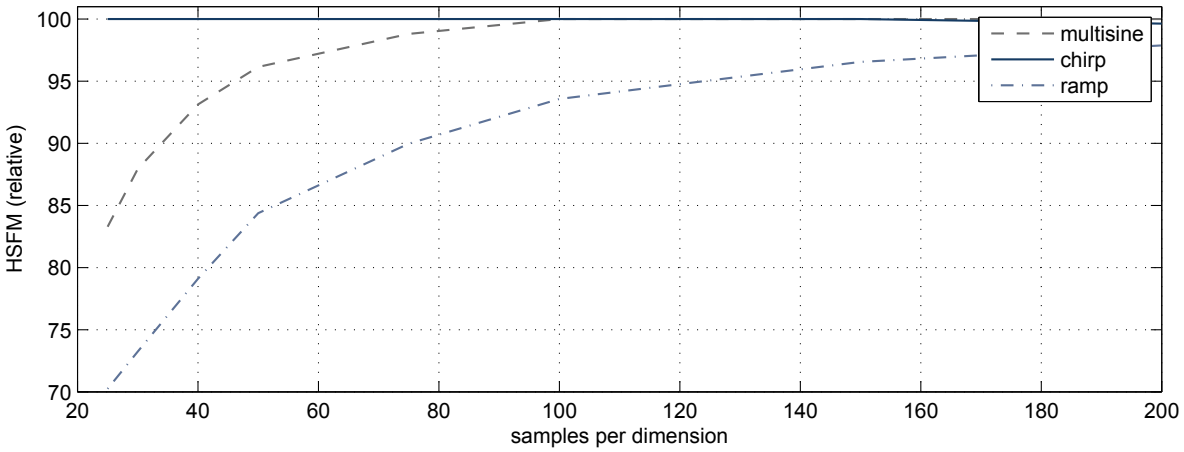


Figure 4.22: HSFM relative for the stationary model inputs and the output (four dimensions)

Also the dynamical values can be analysed, see Fig. 4.23. However, though the ramp DoE initially showed the best space filling property (see Fig. 4.19) the measurement of the chirp DoE is beneficial compared to multisine and ramp DoE. The benefit of the shifted chirp signal regarding space filling property was discussed in chapter 2.4.4. The chirp signals are constantly shifted in a way such that all combinations of the excitation frequencies can be obtained, if the signals are repeated sufficiently many times. This benefit can be seen for the stationary data (see Fig. 4.22) and also for the dynamical data (see Fig. 4.23). Regarding these results it can be assumed that a model which is trained by the ramp DoE measurement will be not able to simulate the chirp DoE measurement.

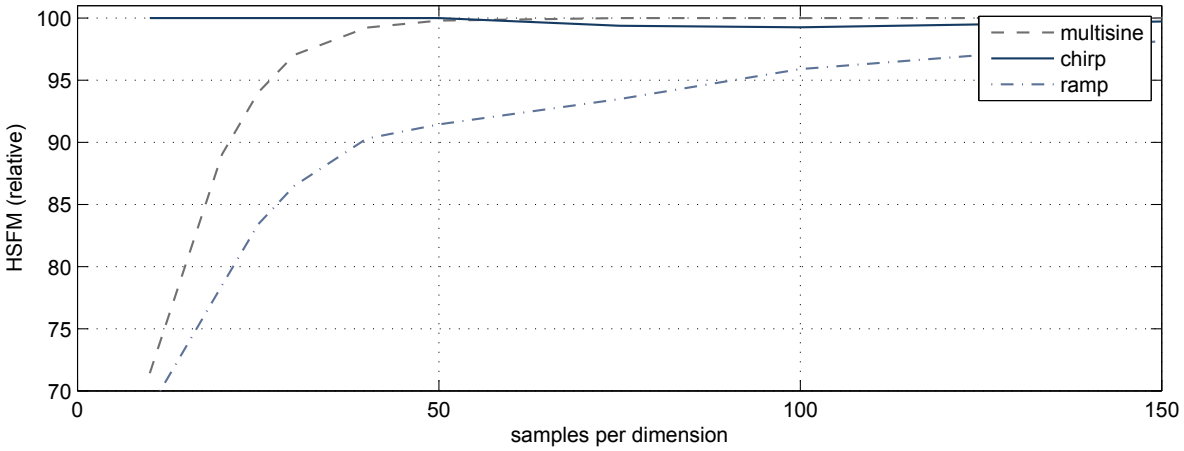


Figure 4.23: HSFM relative for the dynamical model inputs and the output (seven dimensions)

The results of the modelling step confirm the assumption. Results of ETAS ASCMO with a sampling rate of 0.02 seconds and approximately 30000 training points:

DoE plan	R^2 Ramps	R^2 Chirp	R^2 Multisine
Ramps (605 seconds)	0.99[1]	0.62	0.93
Chirp (605 seconds)	0.867	0.98[1]	0.89
Multisine (605 seconds)	0.92	0.814	0.98[1]

[1] training result

GPR results using a HSFM downsampling to 2000 points:

DoE plan	R^2 Ramps	R^2 Chirp	R^2 Multisine
Ramps (605 seconds)	0.96[1]	0.53	0.937
Chirp (605 seconds)	0.79	0.96[1]	0.84
Multisine (605 seconds)	0.96	0.84	0.99[1]
All data	0.96[2]	0.97[2]	0.96[2]

[1] training result
[2] training data inside

As can be seen, in both cases the model which is trained by the ramp or multisine DoE measurement is not able to simulate the chirp measurement well. Furthermore, the sparse GPR shows better results for the ramps and the chirps. The HSFM downsampling also shows good results and especially the multisine can be modelled very well. The HSFM allows all training data to be merged together and downsample again to 2000 training points. The result is given in the last raw *All data*. Thus, merging the data and using HSFM downsampling seems beneficial. In Fig. 4.24 the results of the model using all data and HSFM downsampling is shown for the last 100 seconds of the different DoE measurements.

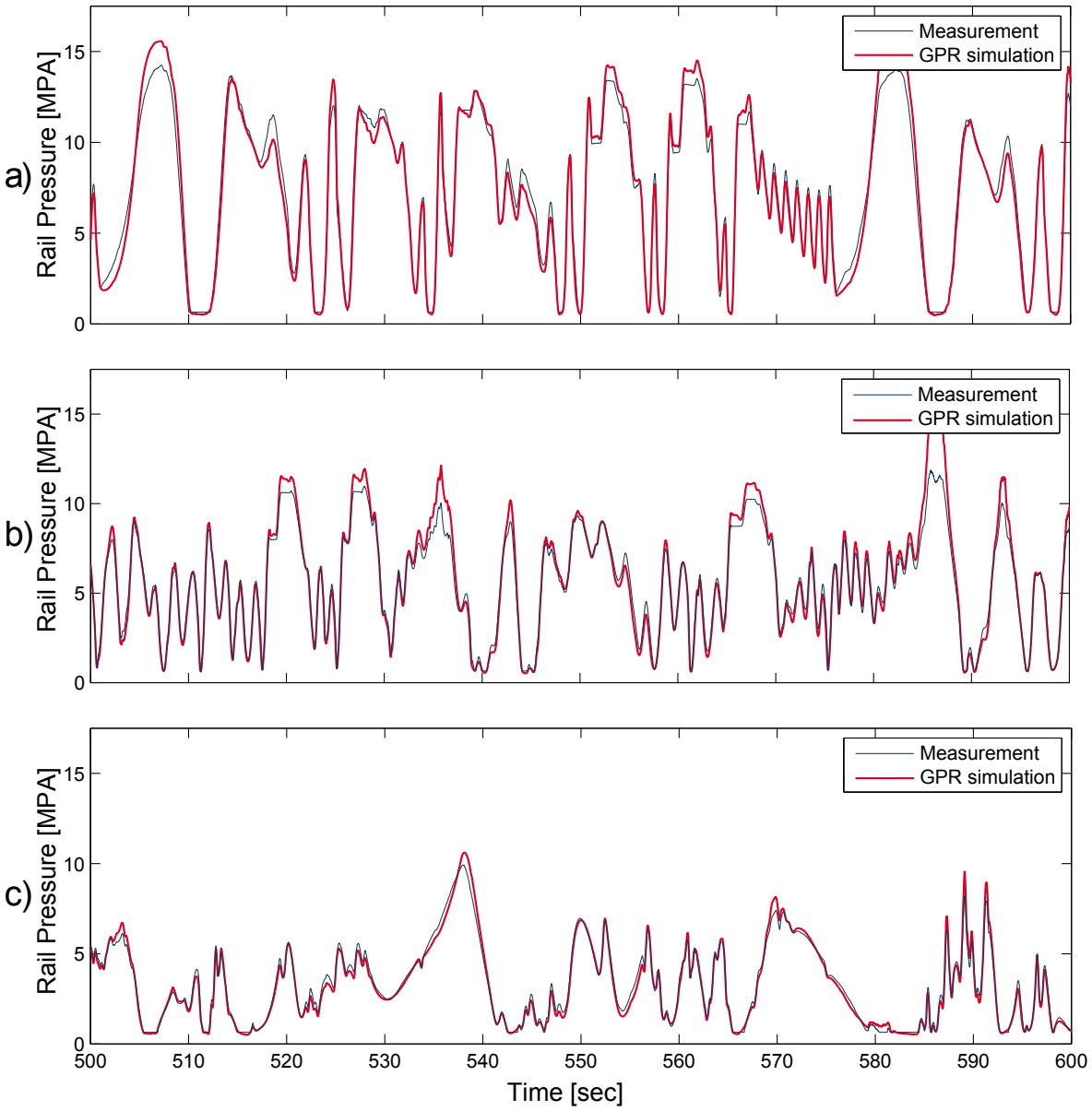


Figure 4.24: GPR simulation results using all data and HSFM downsampling to 2000 training points for the last 100 seconds: a) chirps, b) multisine and c) ramps

4.2.3 Model-based calibration of the HPFS system

The next step of using this model and especially to achieve a benefit with this model is challenging. The main target is to fill the inputs of the dynamical model with realistic inputs. Since these inputs are often directly affected by the modelled system, a closed loop simulation is needed. Here, the most convenient way is to use a Hardware-

in-the-Loop (HiL) system since these systems are usually available in the software development process of ECU. Fig. 4.25 illustrates an overview of a HiL environment.

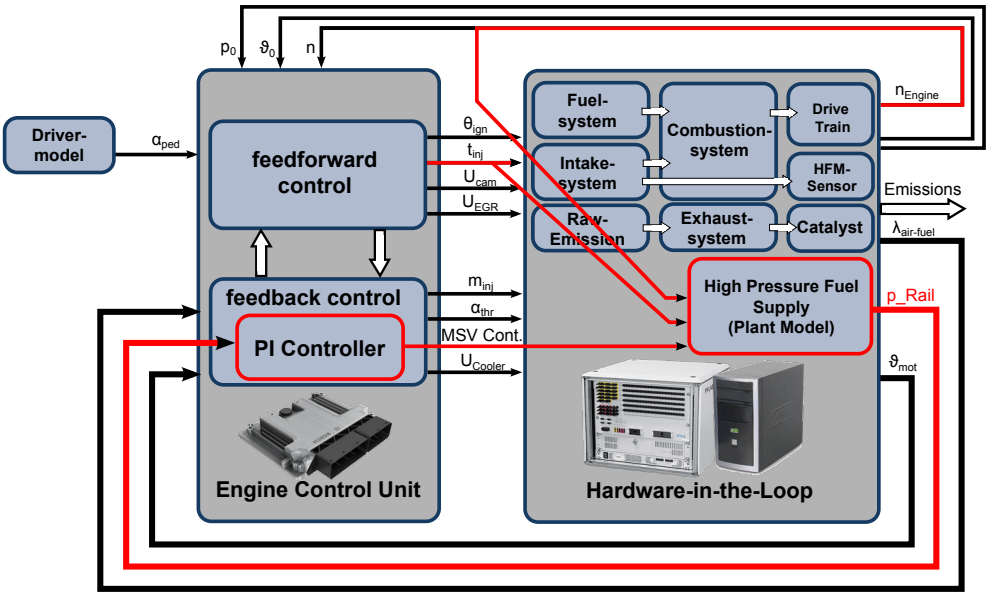


Figure 4.25: HiL environment for closed-loop simulation (Tietze et al. 2014a)

The implementation of the new model is a difficult step, since the model inputs are also simulated and thus have to be verified. In this example, the input of injection time t_{inj} is affected by the lambda controller. To ensure realistic injection times, the simulation of lambda has to be checked and thus the simulation models of combustion, the air system and the fuel system have to be parametrised well (see Fig. 4.26). For example in order to receive realistic controller behaviour of the lambda controller, the positions and dynamics of the oxygen sensors have to be simulated. This means for the after treatment sub models that the the dead time and the delay time have to be adapted. Therefor step response measurements can be used. Since the dead time and the delay time of the exhaust gas is depending of the engine load and the engine speed, the step response measurements can be done by measuring a step of the injection at a defined operating point (constant engine load and engine speed). A detailed overview of the model parametrisation is given in Tietze (2011).

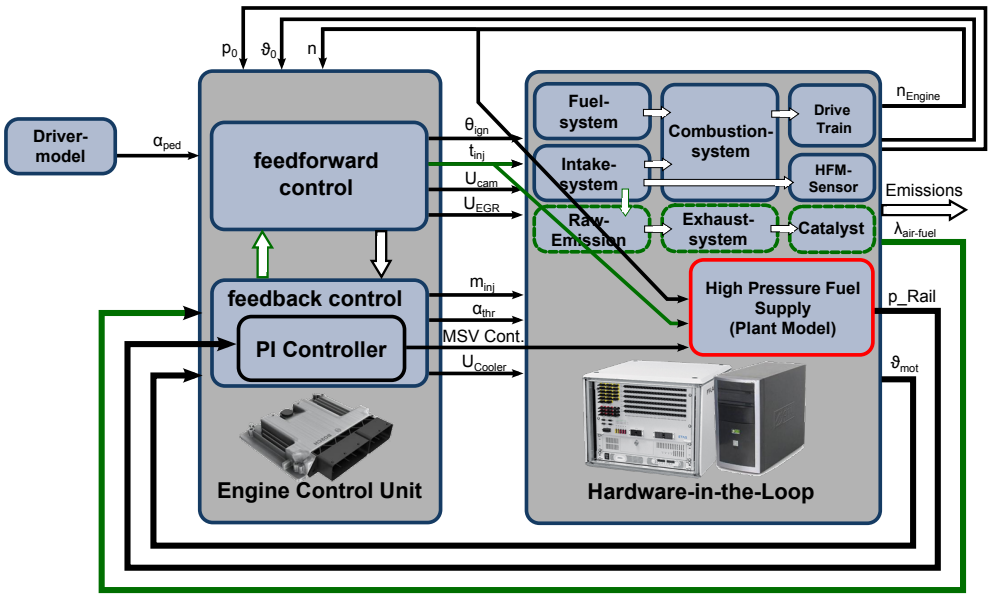


Figure 4.26: Parametrisation of sub models

After validating the HiL environment, the frequency response measurement can be done, see Appendix A.4.2. By exciting with periodic test signals, the so-called *Frequency Response Measurement* allows the determination of the relevant frequency range for linear systems for certain discrete points in the frequency spectrum (Isermann 2011). In Fig. 4.27 the standard FRM method for ECU controller functions of real test vehicles is depicted. In ① multisine excitation signals are measured for different operating points where the engine speed is kept constant. The particular multisine is shown in ②. The periodic system response is measured (see ③). By computing the FFT of the input and the output signals, the Bode plot can be generated and be directly used to estimate the phase and gain margins (see ④).

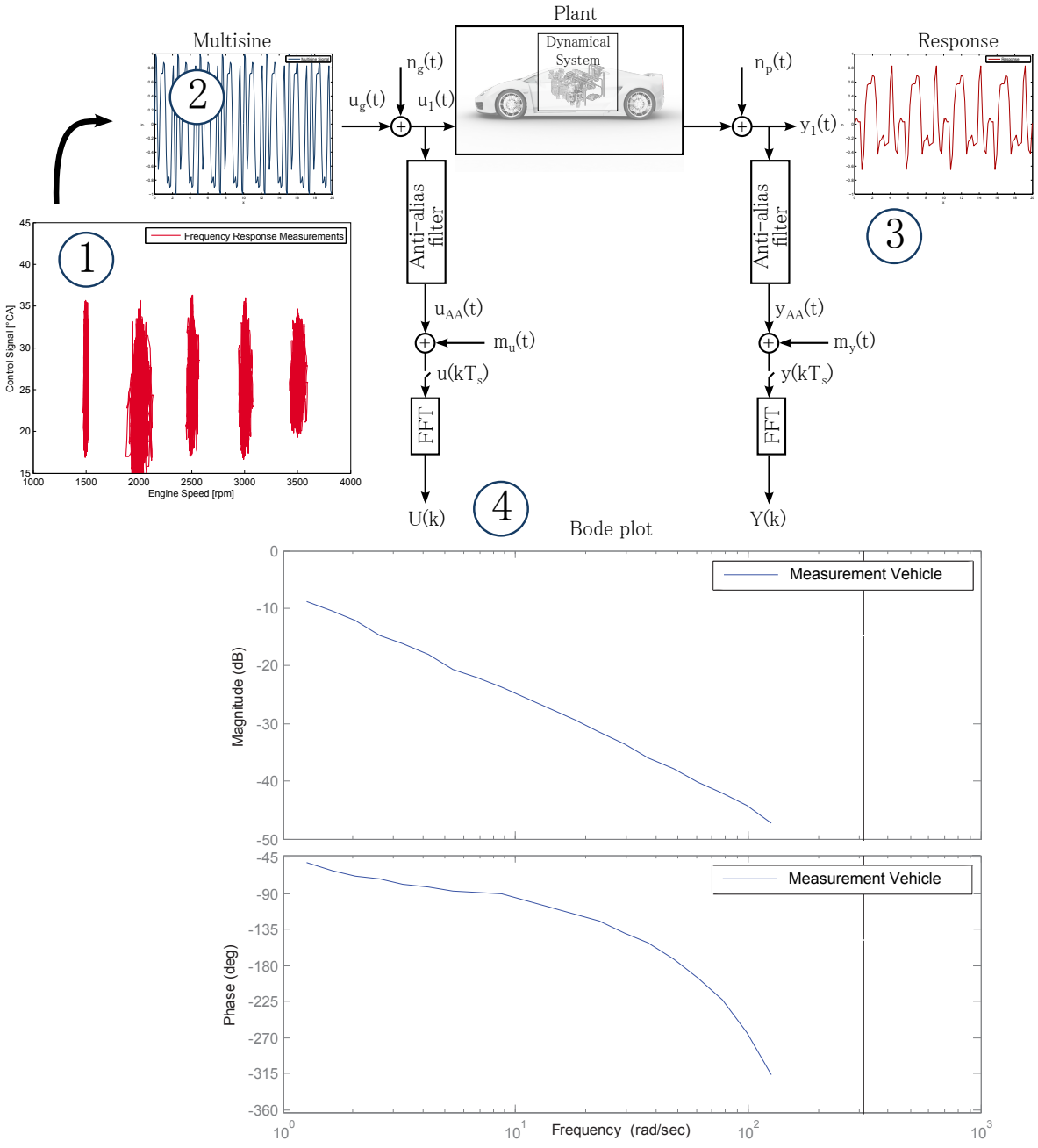


Figure 4.27: FRM of the system at the real test vehicle

In the first step the GPR model can be validated by these mulstisine measurements of the real test vehicle. The offline simulation results are shown in Fig. 4.28. It can be seen that the model is able to simulate the system with high accuracy.

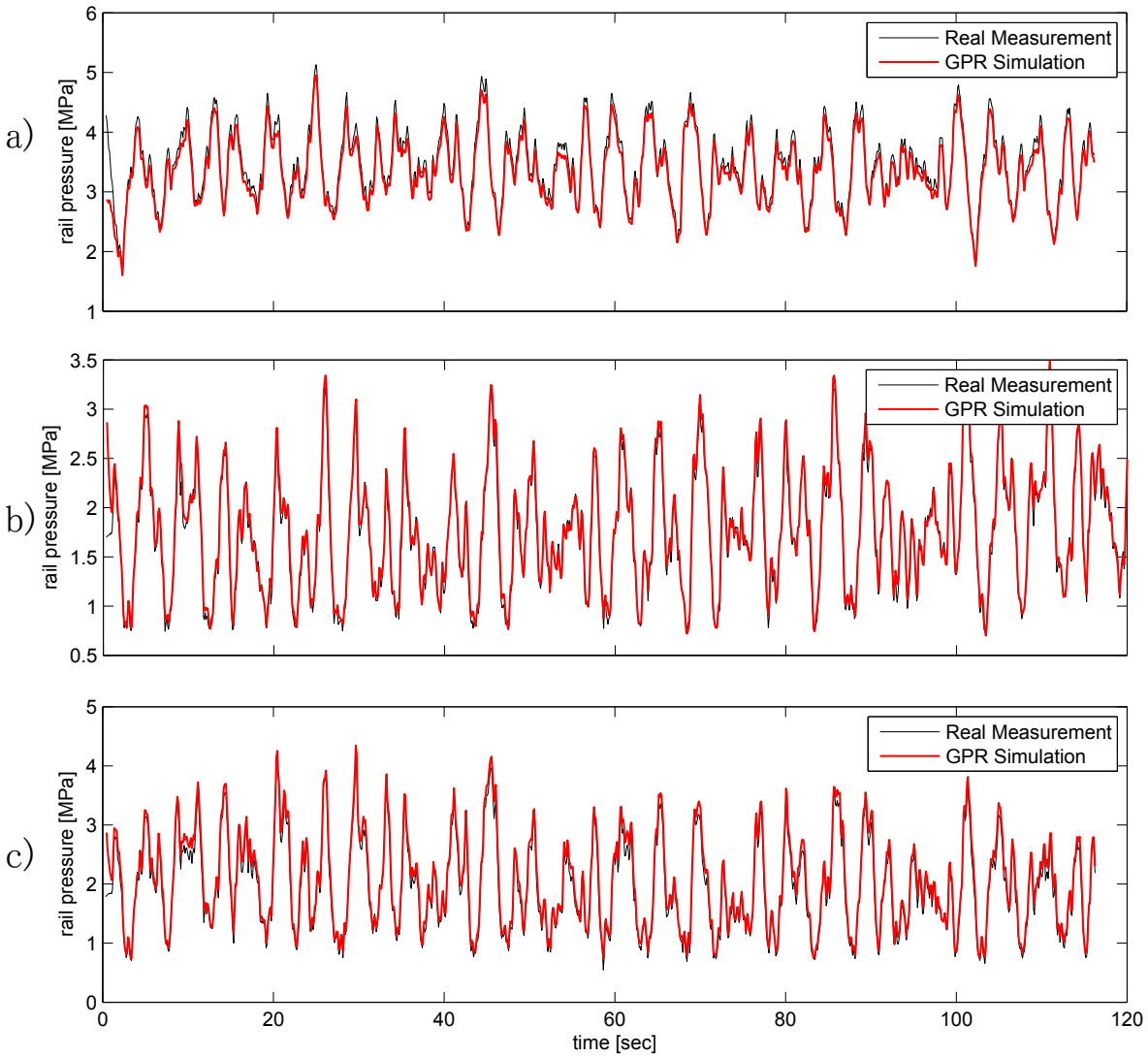


Figure 4.28: GPR model validation for multisine measurements: a) for 1500 rpm, b) 2000 rpm and c) 2500 rpm

Instead of using the real test vehicle the FRM can be used for model-based calibration. In Fig. 4.29 the equivalent procedure is shown for the virtual test vehicle (see ①). In ② the input space coverage of the chirp DoE is shown. It is important to choose the multisine amplitudes with regard to the measured DoE in order to prevent extrapolation.

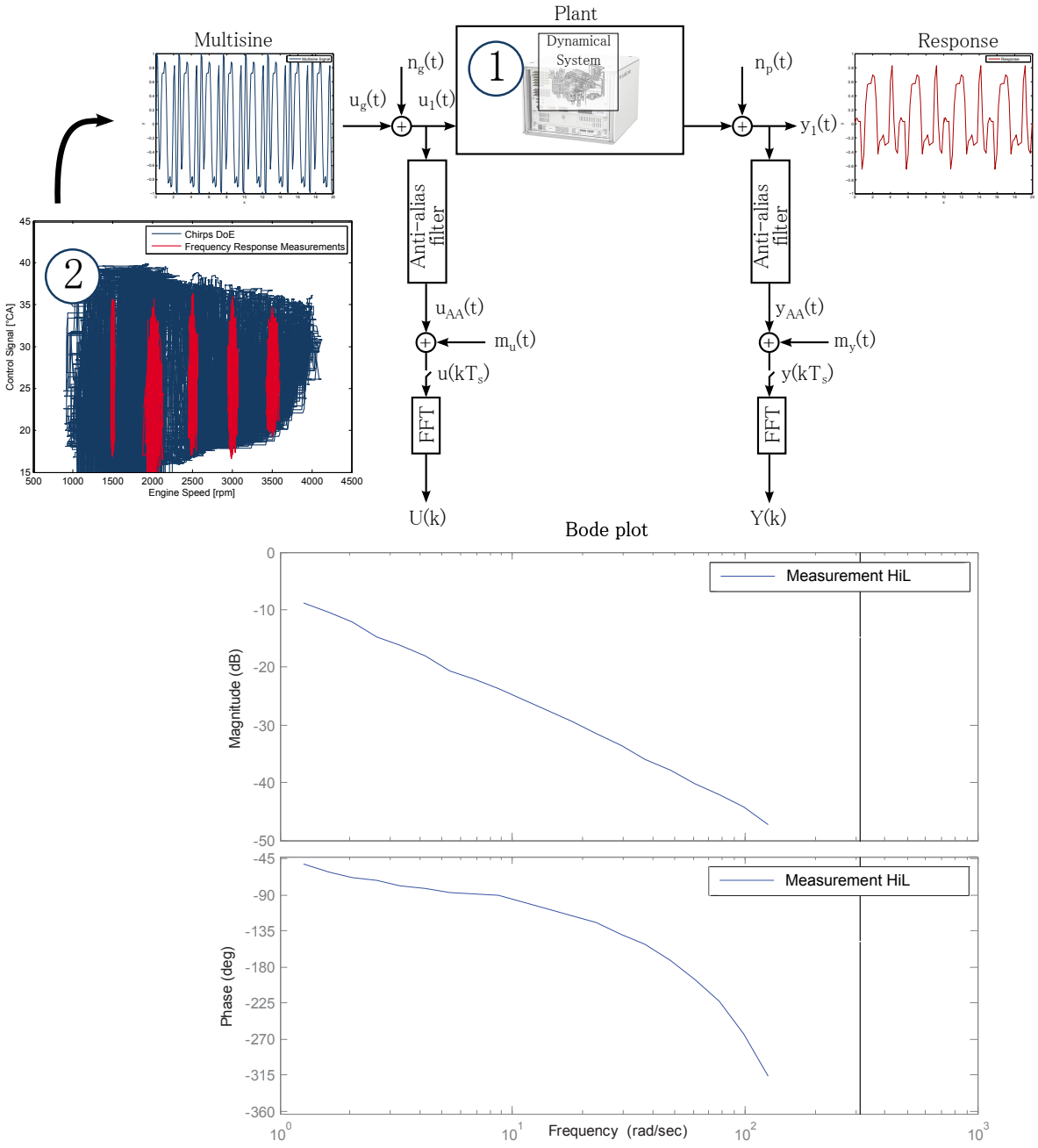


Figure 4.29: FRM of the system at the virtual test vehicle

The comparison of the Bode plots are shown in Fig.4.30. Here, in the first step the measurement at the vehicle was used as input for the dynamic model to give a first indication of the model result. Of course it is not the focus to use a real measurement for an Open-Loop simulation, since the real car is still needed. Thus in the second step the HiL system was used to generate the inputs for the dynamical model. It can

be seen that the Bode plots are comparable and that a model-based calibration of the HPFS controller would be possible.

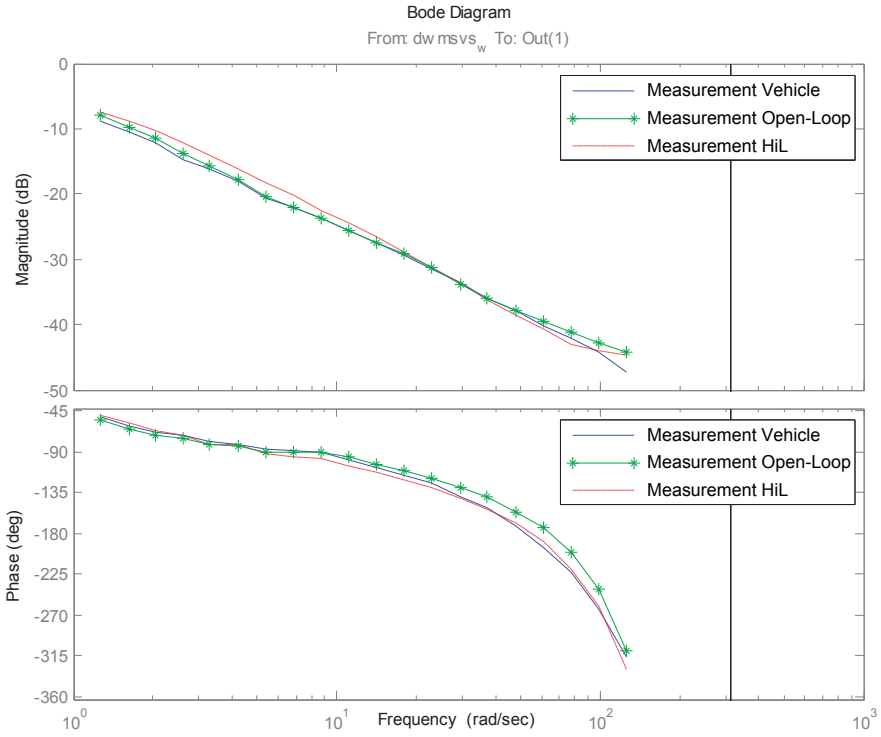


Figure 4.30: Comparison of Bode diagram for frequency response measurements (Tietze et al. 2014a)

5 Summary and Outlook

This thesis deals with data-based modelling techniques as an attractive alternative to physical models to increase efficiency in the modelling process for model-based ECU calibration. Since the ECU calibration process requires a dynamical system response, an external dynamic approach is used to give the stationary regression models a dynamical structure. Due to this structure, the requirements for the dynamical Design of Experiment are analysed in order to obtain adequate system measurements. Four signal types are analysed to determine their suitability for system excitation. APRBS, Ramps, Chirps and Multisine are introduced. A method is presented on how to scale the signals into the permissible input boundaries and a metric is introduced in order to get information about the dynamical input space coverage. Furthermore, a concept for online-DoE measurements is presented.
However, it becomes clear that the dynamic DoE step is not straightforward and will not lead to perfect system measurement covering all dynamical system states. Thus, a flexible modelling algorithm is needed which allows iterative modelling. A model comparison of different algorithms suggests using a combination of LOLIMOT and GPR algorithm in order to receive the iterative modelling strategy with high modelling accuracy. Local Gaussian Process Regression is introduced and modified to a *variance weighted LGPR*, which allows a combination of GPR models and is easy to use. VW-LGPR is well-suited for the identification of dynamical systems, since validation measurements can be used to recalibrate the GPR model.
A real-world application shows the benefit of VW-LGPR compared to the standard method. It shows that VW-LGPR has advantages due to the number of required training data, especially if local effects appear. However, it becomes clear that data clustering is the key step in order to use the LGPR approach successfully. In the second application the High Pressure Fuel Supply control system is identified and the dynamical plant model is implemented into a closed-loop environment. It is shown that the HPFS ECU controller function can be calibrated model-based by frequency response measurements.

The model-based calibration of ECU functions using HiL systems is a costly task since a complete vehicle model needs to be calibrated. From the economical point of view more ECU calibration applications have to be transferred to the virtual vehicle in order to increase the ratio of the usage and the investment. The vision of model-based calibration is to replace the real ECU with a software model. The so-called Software-in-the-Loop environment does not require expensive hardware and models faster than real-time. Thus, the cheap usage of the SiL systems allows optimisation which leads to better and faster ECU function calibration (see Fig.5.1).

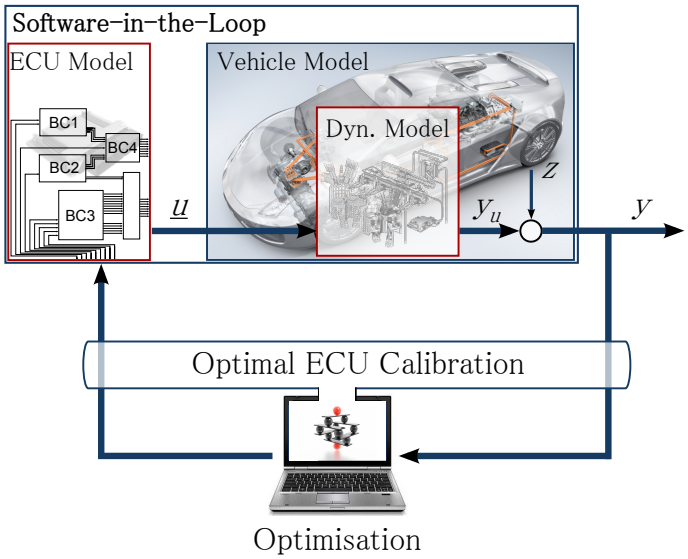


Figure 5.1: Vision: optimal calibration using Software-in-the-Loop and optimisation strategies (Tietze et al. 2014a)

A Appendix

A.1 Mathematical Backround

A.1.1 Probability theory

Let x be a *random variable* having probability density $p(x)$. Its *mean*, *variance* and *second moment* are defined by the expectation values (Orfanidis 1996).

$$\begin{aligned} m = E[x] &= \int_{-\infty}^{\infty} xp(x)dx = \quad \text{mean} \\ \sigma^2 = var(x) = E[(x-m)^2] &= \int_{-\infty}^{\infty} (x-m)^2 p(x)dx = \quad \text{variance} \\ E[x^2] &= \int_{-\infty}^{\infty} x^2 p(x)dx = \quad \text{second moment} \end{aligned}$$

These quantities are known as *second-order statistics* of the random variable x (Orfanidis 1996). The probability density is always normalised to unity by

$$\int_{-\infty}^{\infty} p(x)dx = \quad 1$$

The *Probability Density Function* (PDF) of a univariate Gaussian distribution is given as:

$$p(x) = \mathcal{N}(x \mid m, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2\sigma^2}(x-m)^2\right]$$

An example to the Gaussian PDF is given in Fig. A.1

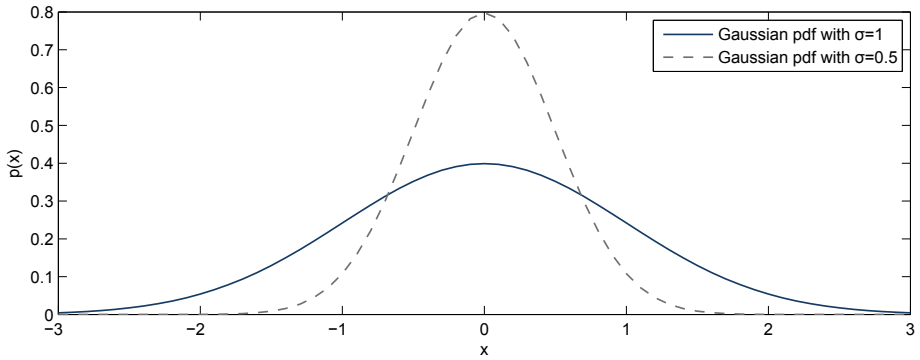


Figure A.1: pdfs for zero mean Gaussians with $\sigma = 1$ and $\sigma = 0.5$

The PDF of a multivariate Gaussian distribution is given as:

$$\begin{aligned} p(\boldsymbol{x}) &= \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) \\ &= \frac{1}{(2\pi)^{N/2}(det(\Sigma)^{1/2})} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right] \end{aligned}$$

Here, $\boldsymbol{x}$ is a Gaussian random vector $\boldsymbol{x} = [x_1, x_2, ...x_N]^T$. Thus, a multivariate Gaussian is completely specified by its mean $\boldsymbol{\mu}$ and its *covariance matrix* Σ:

$$\begin{aligned} \boldsymbol{\mu} &= E[\boldsymbol{x}] \\ \Sigma &= E[(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^T] \end{aligned}$$

Gaussian Approximation for LGPR:
The GP-mean is given as:

$$E[y \mid x] = \int y\, \mathcal{N}\left(y \mid \mu(x), \sigma^2(x)\right) dy = \mu(x) \tag{A.1}$$

and the GP-variance as:

$$\begin{aligned} E[(y - E[y \mid x])^2 \mid x] &= E[y^2 \mid x] - (E[y \mid x])^2 \\ &= \int y^2 \mathcal{N}(y \mid \mu(x), \sigma^2(x))dy - \left(\int y\, \mathcal{N}(y \mid \mu(x), \sigma^2(x))dy\right)^2 \\ \sigma^2(x) &= \int y^2 \mathcal{N}(y \mid \mu(x), \sigma^2(x))dy - (\mu(x))^2 \end{aligned} \tag{A.2}$$

the approximation of the GPs for the LGPR model can be done by:

$$p(y \mid x) = \sum_{k=1}^{N} \pi_k(x)\mathcal{N}(y \mid \mu_k(x), \sigma_k^2(x)) \quad \text{with} \quad \pi_k(x) = \frac{\sigma_k^2(x)^{-1}}{\sum_{k=1}^{N} \sigma_k^2(x)^{-1}}$$

$$0 \leq \pi_k(x) \leq 1 \quad \sum_{k=1}^{N} \pi_k(x) = 1$$

Plugging this into Eq. A.1 gives the-LGPR mean:

$$\begin{aligned} m(y \mid x) &= \int y \; p(y \mid x) dy = \int y \sum_{k=1}^{N} \pi_k(x) \mathcal{N}(y \mid \mu_k(x), \sigma_k^2(x)) dy \\ &= \sum_{k=1}^{N} \pi_k(x) \int y \; \mathcal{N}(y \mid \mu_k(x), \sigma_k^2(x)) dy \\ &= \sum_{k=1}^{N} \pi_k(x) \mu_k(x) \end{aligned} \tag{A.3}$$

The LGPR-variance can now be computed by using Eq. A.2 and Eq. A.3:

$$\begin{aligned} v(y \mid x) &= E[y^2 \mid x] - (E[y \mid x])^2 \\ &= \int y^2 p(y \mid x) dy - \left(\sum_{k=1}^{N} \pi_k(x) \mu_k(x) \right)^2 \\ &= \int y^2 \sum_{k=1}^{N} \pi_k(x) \; \mathcal{N}(y \mid \mu_k(x), \sigma_k^2(x)) dy - \left(\sum_{k=1}^{N} \pi_k(x) \mu_k(x) \right)^2 \\ &= \sum_{k=1}^{N} \pi_k(x) \int y^2 \; \mathcal{N}(y \mid \mu_k(x), \sigma_k^2(x)) dy - \left(\sum_{k=1}^{N} \pi_k(x) \mu_k(x) \right)^2 \end{aligned} \tag{A.4}$$

Now the integral is given by Eq. A.2:

$$\int y^2 \mathcal{N}(y \mid \mu(x), \sigma^2(x)) dy = \sigma^2(x) + (\mu(x))^2$$

Plugging this into Eq. A.4 finally gives the variance:

$$v(y \mid x) = \sum_{k=1}^{N} \pi_k(x) (\sigma_k^2(x) + (\mu_k(x))^2) - \left(\sum_{k=1}^{N} \pi_k(x) \mu_k(x) \right)^2$$

Product of Gaussian distributions:
Given $\boldsymbol{y} \in \mathbb{R}^n$ as a multivariate normal distributed vector of random variables whose mean depends linearly on $\boldsymbol{f} \in \mathbb{R}^m$ and $\boldsymbol{P} \in \mathbb{R}^{n \times m}$, then the product:

$$\mathcal{N}(\boldsymbol{y} \mid \boldsymbol{P}\boldsymbol{f}, \boldsymbol{B}) \times \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{a}, \boldsymbol{A}) \propto \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{A.5}$$

is again proportional to a multivariate normal density with mean and covariance

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}(\boldsymbol{A}^{-1}\boldsymbol{a} + \boldsymbol{P}^T\boldsymbol{B}^{-1}\boldsymbol{y}) \ \in \mathbb{R}^m \tag{A.6}$$

and covariance matrix

$$\boldsymbol{\Sigma} = (\boldsymbol{A}^{-1} + \boldsymbol{P}^T\boldsymbol{B}^{-1}\boldsymbol{P})^{-1} \ \in \mathbb{R}^{m\times m} \tag{A.7}$$

where $\boldsymbol{a} \in \mathbb{R}^m$, the covariance matrix $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ Kuss (2006).

Integral of two Gaussian:

$$\int_{\mathbb{R}^m} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{a}, \boldsymbol{A})\mathcal{N}(\boldsymbol{a} \mid \boldsymbol{b}, \boldsymbol{B}) \ d\boldsymbol{a} = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{b}, \boldsymbol{A} + \boldsymbol{B}) \tag{A.8}$$

Marginal and Conditional distributions:
Let $x \sim \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ be partitioned $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]^T$ such that

$$\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{bmatrix} \mid \begin{bmatrix}\boldsymbol{\mu}_1\\ \boldsymbol{\mu}_2\end{bmatrix} \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{bmatrix}\right) \tag{A.9}$$

then the marginal distributions are

$$\boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) \tag{A.10}$$
$$\boldsymbol{x}_2 \sim \mathcal{N}(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{A.11}$$

And the conditional distributions are:

$$\boldsymbol{x}_1 \mid \boldsymbol{x}_2 \sim \mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}) \tag{A.12}$$
$$\boldsymbol{x}_2 \mid \boldsymbol{x}_1 \sim \mathcal{N}(\boldsymbol{x}_2 \mid \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1), \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}) \tag{A.13}$$

A.1.2 Completing the squares

Given the Bayesian linear regression problem of chapter 3.3.2.2

$$
\begin{aligned}
&Posterior = Likelihood \times Prior \\
p(\boldsymbol{w} \mid \mathcal{D}) &\propto \exp(-\frac{a}{2}|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}|^2)\exp(-\frac{b}{2}\boldsymbol{w}^T\boldsymbol{w}) \\
&\propto \exp(-\frac{a}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) - \frac{b}{2}\boldsymbol{w}^T\boldsymbol{w}) \\
&\propto \exp(a(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + b\boldsymbol{w}^T\boldsymbol{w}) \\
&\propto \exp(a(\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w}) + b\boldsymbol{w}^T\boldsymbol{w}) \\
&\propto \exp(a\boldsymbol{y}^T\boldsymbol{y} - 2a\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + a\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} + b\boldsymbol{w}^T\boldsymbol{w}) \\
&\propto \exp(a\boldsymbol{y}^T\boldsymbol{y} - \underbrace{2a\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y}}_{termB1} + \underbrace{\boldsymbol{w}^T(a\boldsymbol{X}^T\boldsymbol{X} + b\boldsymbol{I})\boldsymbol{w}}_{termA1}
\end{aligned}
\tag{A.14}
$$

Given a multivariate Gaussian with Λ as the inverse of the covariance matrix:

$$
\begin{aligned}
\mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \Lambda) &\propto \exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{\mu})^T\Lambda(\boldsymbol{w} - \boldsymbol{\mu})\right) \\
&\propto \exp\left(-\frac{1}{2}(\underbrace{\boldsymbol{w}^T\Lambda\boldsymbol{w}}_{termA2} - \underbrace{2\boldsymbol{w}^T\Lambda\boldsymbol{\mu}}_{termB2} + \boldsymbol{\mu}^T\Lambda\boldsymbol{\mu})\right)
\end{aligned}
\tag{A.15}
$$

To bring Eq. A.14 into the form of Eq. A.15 the $\boldsymbol{w}$ dependant terms are compared, here A1 with A2 and B1 with B2. The comparison of A1 with A2 directly gives:

$$
\boxed{\Lambda = a\boldsymbol{X}^T\boldsymbol{X} + b\boldsymbol{I}} \tag{A.16}
$$

The comparison of B1 and B2 gives:

$$
\begin{aligned}
2a\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} &= 2\boldsymbol{w}^T\Lambda\boldsymbol{\mu} \\
a\boldsymbol{X}^T\boldsymbol{y} &= \Lambda\boldsymbol{\mu}
\end{aligned}
$$

Assuming that Λ is invertible:

$$
\boxed{\boldsymbol{\mu} = a\Lambda^{-1}\boldsymbol{X}^T\boldsymbol{y}} \tag{A.17}
$$

Finally the multivariate Gaussian is given as:

$$p(\boldsymbol{w} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{w} \mid \boldsymbol{\mu}, \Lambda) \tag{A.18}$$

A.2 Data Clustering

When using the k-means cluster algorithm the user has to define a fixed number of clusters. K-means clustering minimieses the following loss function:

$$I = \sum_{j=1}^{C} \sum_{i \in \mathcal{S}_j} \|\boldsymbol{u}(i) - \boldsymbol{c}_j\|^2 \rightarrow \min_{\boldsymbol{c}_j} \tag{A.19}$$

where the index i runs over all elements of the sets $\mathcal{S}_j$, C is the number of clusters, and $\boldsymbol{c}_j$ are the cluster centers. The sets $\mathcal{S}_j$ contain all indices of those data samples that belong to the cluster j, i.e., which are nearest to the cluster center $\boldsymbol{c}_j$. The cluster centers $\boldsymbol{c}_j$ are the parameters that the clustering technique varies in order to minimize Eq. A.19 (Nelles 2001).

A.3 MATLAB Codes

A.3.1 HSFM Algorithm

```
%%---Descrpition---%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This algortihm computes a heuristic metric regarding the property of space filling.
%
%%%---Usersettings---%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
samp_per_dim=5;                         % samples per dimension
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

X=get(measurement);                     %meas is the measuremnet array (Lxd)
                                        %   L=number of sample points,
                                        %   d=dimension of inputs + output
d=size(meas.Data,2);                    %number of dimensions
theo_samp=samp_per_dim^d;               %theoretical number of samples
                                        %   without constraints
for i=1:size(X,2)                       % normalisation
    X(:,i) = (X(:,i)-min(X(:,i)))./...
    (max(X(:,i))-min(X(:,i))) ;
end
```

```
18
19 D=pdist(X,'euclidean')';                %computes the Euclidean distance
20 Zdist = squareform(D);                  %converts y into a square,
21                                         %   symmetric format Zdist
22 Zdist = triu(Zdist,0);                  %returns the upper triangular part of Zdist
23
24 Zdist_tril=ones(size(Zdist));           %generate matrix with ones of size Zdist
25 Zdist_tril=tril(Zdist_tril,1);          % get the lower triangular part of Zdist
26 Zdist=Zdist+Zdist_tril;                 % get the new Zdist matrix
27
28 [rowIdx,colIdx ]=...
29 find(Zdist<(1/samp_per_dim));           %find the small distances
30
31
32
33 Comb=[rowIdx,colIdx];                   %merge rowIdx and colIdx
34 idx=ones(length(X(:,1)),1);            %create a matrix of ones with
35                                         %   measurement length
36 while size(Comb,1)>1
37     count=histc(Comb,[1:length(Zdist)]);%counts the number of samples that
38                                         %   are close to others
39     count=count(:,1)+count(:,2);        %sum of the counts
40     Max=max(count);                     %get the sample which has the maximum
41                                         %   number of small distances
42     position=find(count==Max);          %get all the positions, where the
43                                         %   maximal value exists
44     idx(position(1))=0;                 %save this position in idx
45     Comb(Comb(:,1)==position(1),:)=[];  %delete the rows with max value
46     Comb(Comb(:,2)==position(1),:)=[];  %delete the rows with max value
47 end
48
49 if ¬isempty(Comb)
50     idx(Comb(1))=0;                     %save the last one of the sample points
51                                         %   with small distance in idx
52 end
53
54 idx_Num=[]; idx_Num=find(idx==1);       %save numbers of idx
55
56
57 for matrixconstruct=2:size(X,2)
58     idx(:,matrixconstruct)=idx(:,1);    %expand the idx vector to a matrix for
59                                         %   multiplication
60 end
61
62 X_HSFM=X.*idx;                          %X_HSFM is the new input space
63 X_HSFM(find(X_HSFM(:,1)==0),:)=[];
64
65 HSFM=(length(X_HSFM))/theo_samp*100;    %HSFM gives the quality metric of the
66                                         %   training data
```

A.4 Signal Processing Backround

A.4.1 Fourier Transform

Generally, a transformation is done to simplify mathematical operations, like logarithm transform. The Fourier transform (FT) is also a universal problem solving technique. The principle of the Fourier transform is shown in Fig. A.2. Given a waveform, the Fourier transform decomposes the waveform into a sum of sinusoids of different frequencies (Brigham 1974).

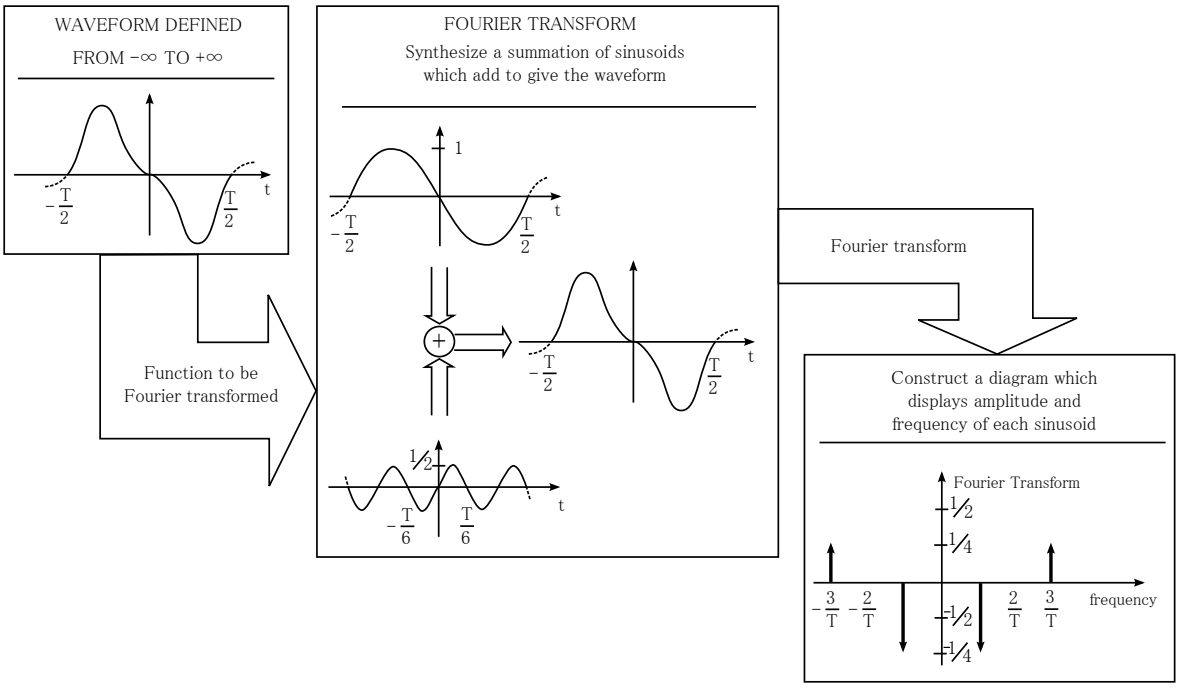


Figure A.2: Interpretation of the Fourier Transform (Brigham 1974)

The Fourier transform identifies the different frequency sinusoids and their respective amplitudes which combine to form an arbitrary waveform (Brigham 1974). Mathematically, this relationship is stated as:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-j2\pi ft} dt$$

where $h(t)$ is the waveform to be decomposed into a sum of sinusoids, $H(f)$ is the Fourier transform of $h(t)$, and $j = \sqrt{-1}$ (Brigham 1974). Typically $h(t)$ is termed a

function of the variable *time* and $H(f)$ is termed a function of the variable *frequency.* In general the Fourier transform is a complex quantity:

$$H(f) = R(f) + jI(f) = |H(f)|e^{j\theta(f)}$$

where $R(f)$ is the *real* part of the Fourier transform,
$I(f)$ is the *imaginary* part of the Fourier transform,
$\mid H(f) \mid$ is the *amplitude* or *Fourier spectrum* of $h(t)$ and is given by $\sqrt{R^2(f) + I^2(f)}$,
$\Phi(f)$ is the *phase angle* of the Fourier transform and is given by $tan^{-1}[I(f)/R(f)]$.
In the following sections some fundamental properties of the Fourier transform are given:

- *Linearity of FT:* If $x(t)$ and $y(t)$ have the Fourier transforms $X(f)$ and $Y(f)$, then the sum $x(t) + y(t)$ has the Fourier transform $X(t) + Y(t)$ (Brigham 1974).

- *Convolution:* The convolution integral is given by

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)dt = x(t) * h(t)$$

  Here $y(t)$ is said to be the convolution of the functions $x(t)$ and $h(t)$ (Brigham 1974). The illustration of the convolution theorem is given in Fig. A.3. The convolution of two rectangular functions (a) and (b) gives a triangular function (e). In the frequency-domain the rectangular functions are represented as $sin(f)/f$ functions (c) and (d). The same result of the convolution in time-domain can be reached by a multiplication in frequency-domain. Thus, the triangular waveform of Fig. A.3 (e) and the $sin^2 f/f^2$ function are Fourier transform pairs.

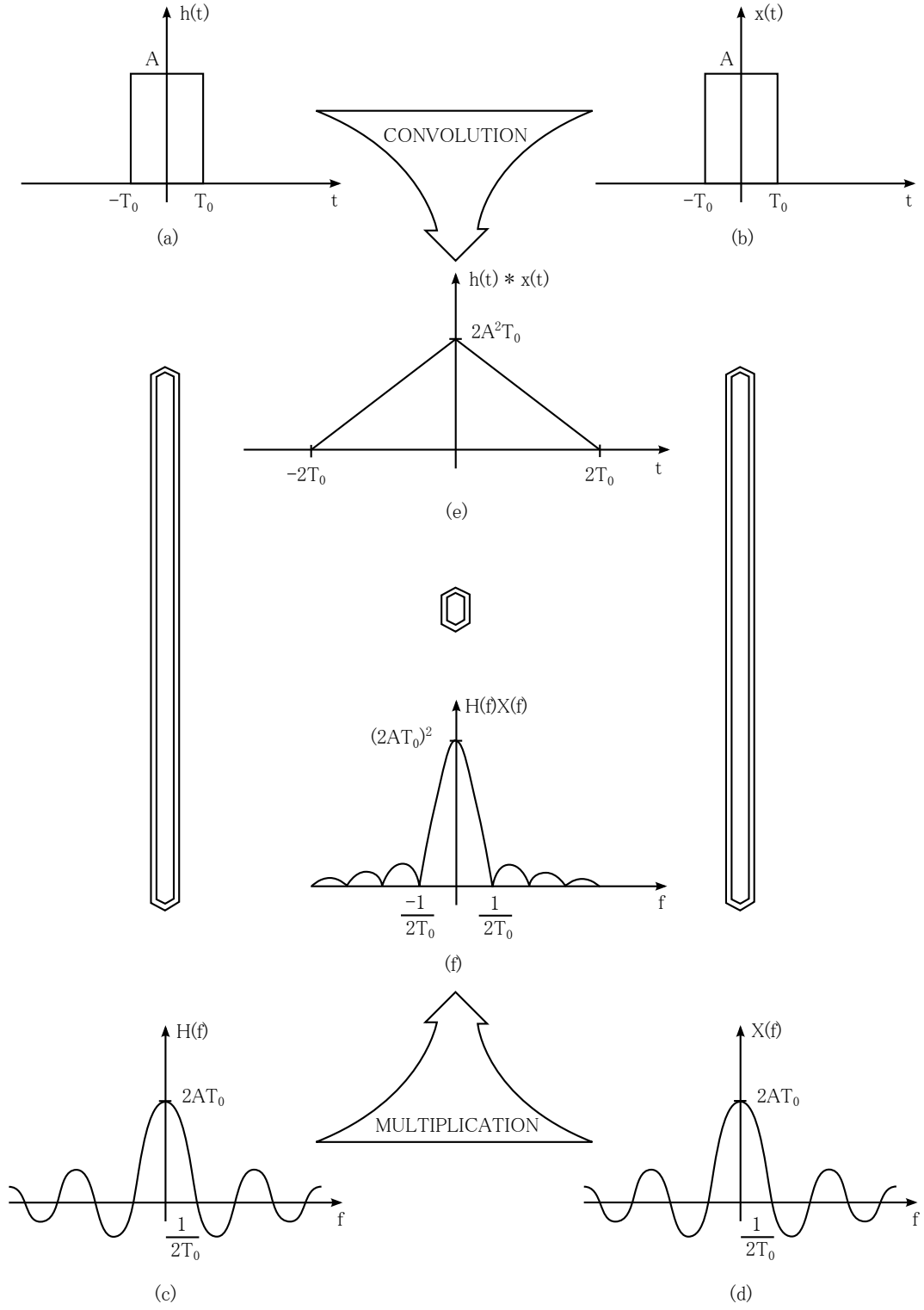


Figure A.3: Graphical example of the convolution theorem (Brigham 1974)

Vice versa, the convolution in frequency domain is equivalent. In Fig. A.4 the

Fourier transform of the product of the cosine waveform (a) and the rectangular waveform (b) is desired. Given the Fourier transform of each signal (c) and (d) the convolution can be easily computed in frequency-domain which yields to (f).

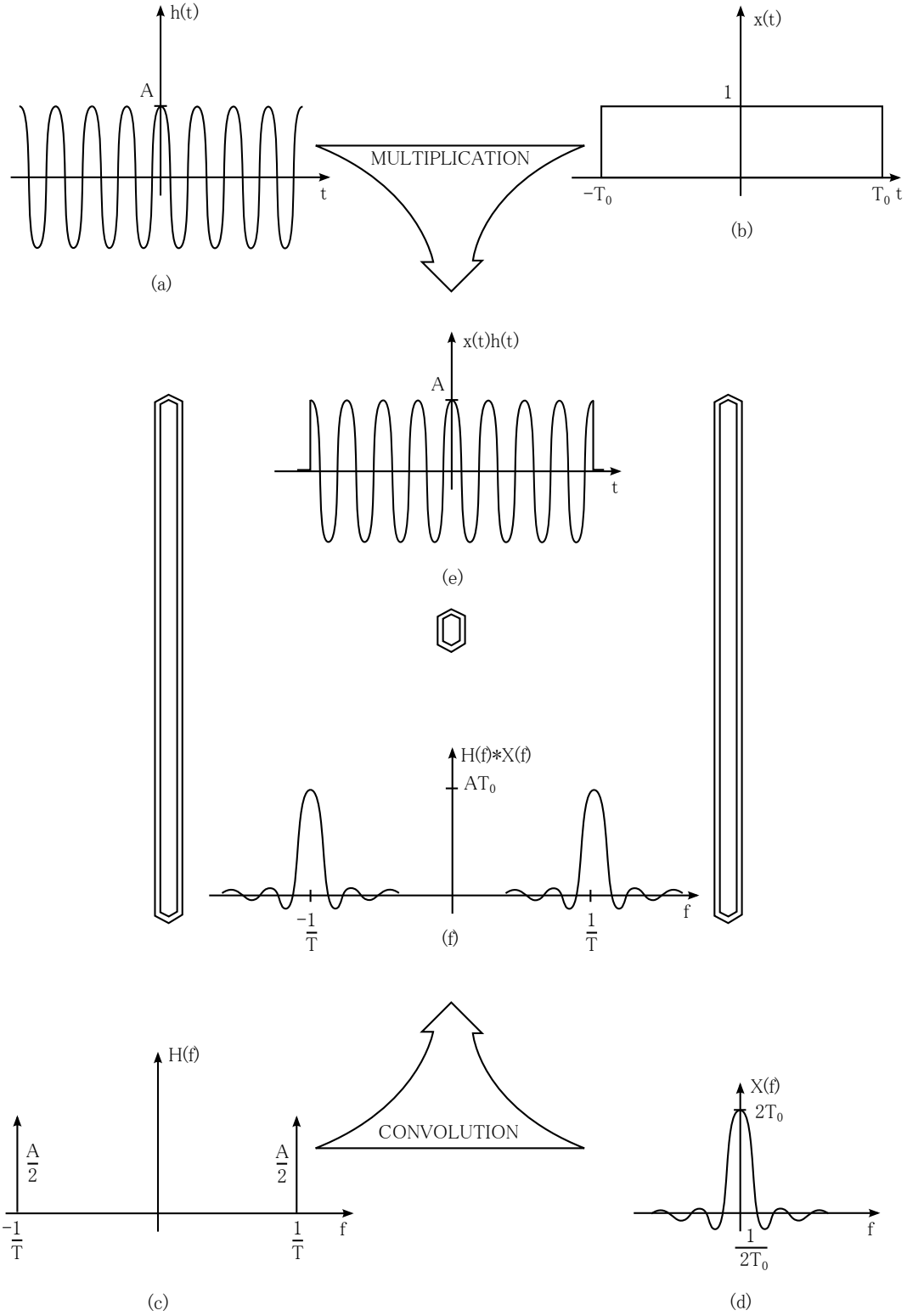


Figure A.4: Graphical example of the frequency convolution theorem (Brigham 1974)

A.4.1.1 Discrete Fourier Transformation

The discretisation and quantisation of continuous-time signals leads to discrete-time signals. Thus a Discrete Fourier Transformation (DFT) is needed to transform the time-domain signal into the frequency-domain. Usually, *Fast Fourier Transformation* (FFT) algorithm is used to efficiently compute the DFT. To measure the spectrum of a continuous time-signal, three basic steps have to be taken (Pintelon & Schoukens 2012):

- *Discretisation in time*: Sample the continuous-time signal in an equidistant time grid. In the time-domain, the sampling process can be formulated as a multiplication with a periodically repeated *Dirac impulse* (Brigham 1974).

$$\tilde{u}_d(t) = u(t)\delta_{T_s}(t) \quad \text{with} \quad \delta_{T_s}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \tag{A.20}$$

- *Windowing*: Restriction of the length of the data to N samples. This can be done by defining a rectangular window of width T given as:

$$w(t) = 1 \quad if \quad 0 \leq t \leq T \quad \text{and} \quad w(t) = 0 \quad \text{elsewhere}$$

  By multiplying this window with the sampled function, a new signal is obtained, differing from zero only in a finite number of samples (Pintelon & Schoukens 2012). Note, that the multiplication in time-domain means a convolution in frequency domain, see section A.2. Thus, the windowing of a signal leads to the effect of *leakage*, see Fig. A.4 sub plot (f). To avoid leakage effects, the window length should be an exact multiple of the period length.

- *Discretisation in frequency*: The FT leads to a continuous frequency signal. Thus, the spectrum needs to be computed at an equidistant set of frequencies. The DFT can be computed as:

$$U_{DFT} = \sum_{n=0}^{N-1} u(nT_s)e^{-j2\pi nk/N}, \quad k = 0, 1, ..., N-1$$

A.4.1.2 DFT for non multiple periods

The DFT delivers an optimal spectrum, if an integer number of periods is measured. However, often this is not possible and leakage effects appear by windowing the signal. If no integer number of periods is given, the aim is to minimise the leakage. This can be done by using alternative windowing functions. Here for instance the *Hanning* or *cosine* window is given as:

$$w(t) = 1 - cos(2\pi t/T) \quad if \quad 0 \leq t \leq T \quad \text{and} \quad w(t) = 0 \quad \text{elsewhere}$$

the aim of all the alternative windows is to taper the signal at the beginning and at the end of the window in order to decrease the discontinuities of the periodically reconstructed signal because they are the basic source of the leakage errors (Pintelon & Schoukens 2012). Fig. A.5 shows the comparison of the rectangular window with the Hanning window.

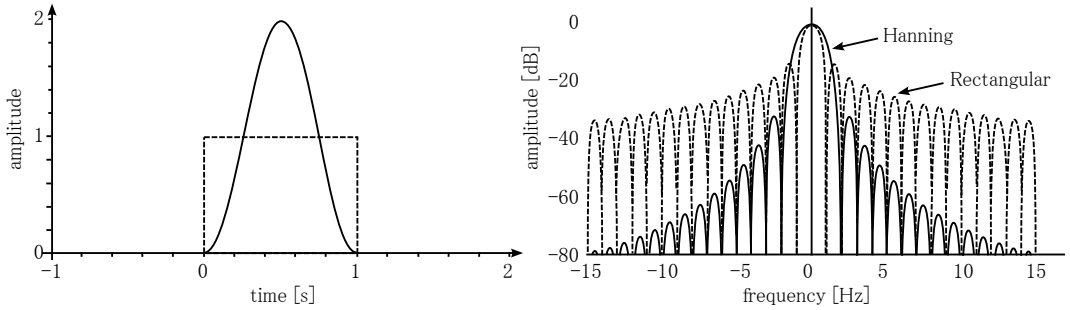


Figure A.5: Comparison of rectangular window with Hanning window in the time-domain (left) and the frequency-doamin (right) (Pintelon & Schoukens 2012)

A.4.2 Frequency Response Measurement

Given a linear dynamic system $G(j\omega)$ between input $u(t)$ and output $y(t)$ as shown in Fig. A.6, the *Frequency Response Function* (FRF) can be used to get a good initial idea about the system under test (Pintelon & Schoukens 2012).

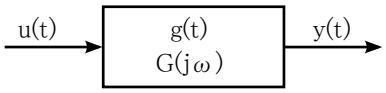


Figure A.6: Block diagramn of linear dynamic system

An FRF consists of transfer function measurements $G(j\omega_k)$ at a discrete set of fre-

quencies $\omega_k, k = 1, ..., F$, where each of these models is nonparametric (Pintelon & Schoukens 2012). In order to achieve the FRF of a linear system the plant must be periodically excited and an integer number of periods of the steady-state response must be measured (Pintelon & Schoukens 2012). The principal measurement setup for an FRF measurement is given in Fig. A.7. The generator signal $r(t)$ is applied to the plant using an actuator $u_g(t)$. The input $u_1(t)$ and output $y_1(t)$ are passed through the anti-alias filter before sampling, resulting in $u_{AA}(t)$ and $y_{AA}(t)$ (Pintelon & Schoukens 2012).

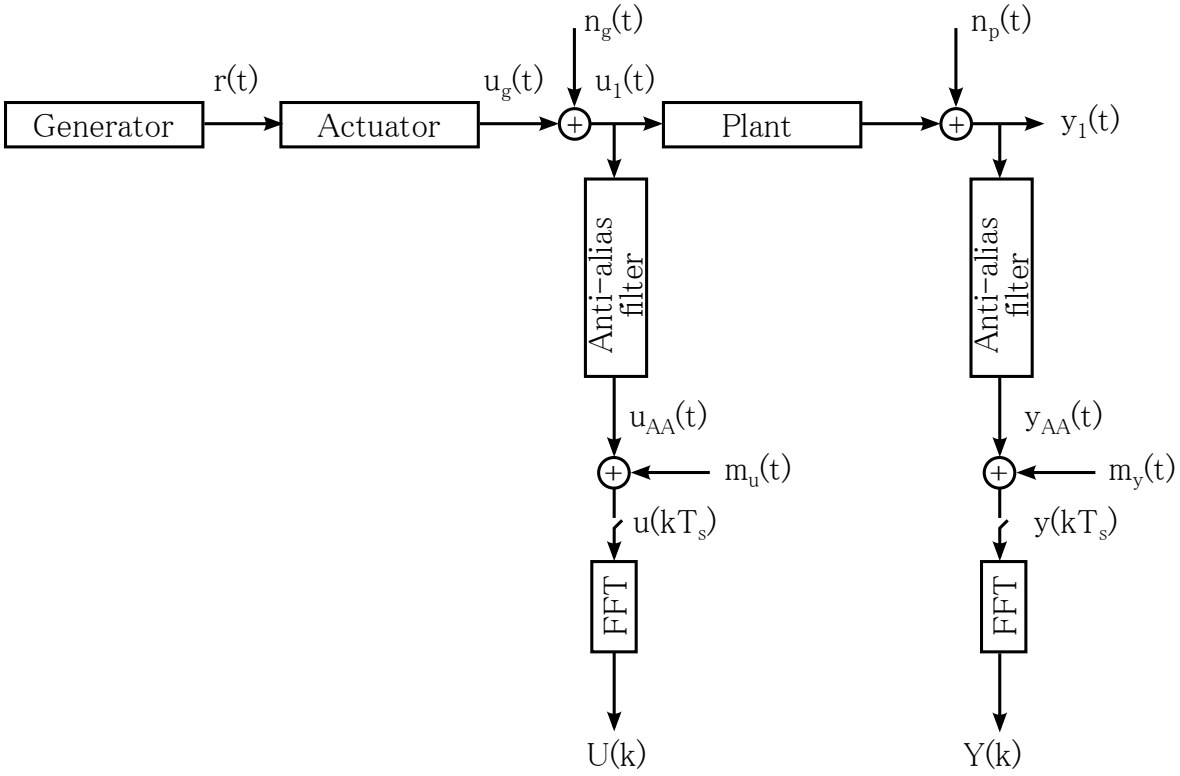


Figure A.7: Principal measurement setup and notation for periodic signals (Pintelon & Schoukens 2012)

These time-domain signals are finally transformed to the frequency-domain using the DFT (see A.2). The FRF at frequency f_k is eventually given as:

$$\hat{G}(j\omega_k) = Y(k)/U(k)$$

The FRF measure is disturbed by noise, see Fig. A.7 i.e. measurement noise $m_u(t)$, output measurement noise $m_y(t)$, system noise $n_p(t)$ and generator noise $n_g(t)$. The

impact of the DFT on the noise is intensively studied in Pintelon & Schoukens (2012). Thus the interested reader is referred to the book of Pintelon & Schoukens (2012).

Nomenclature

API Application-Programming-Interface.

APRBS Amplitude modulated Pseudo Random Binary Sequence.

DFT Discrete Fourier Transformation.

DoE Design of Experiments.

ECU Engine Control Unit.

ESP Electronic Stability Control.

FFT Fast Fourier Transformation.

FRF Frequency Response Function.

FRM Frequency Response Measurement.

FT Fourier transform.

GP Gaussian Process.

GPC Gaussian Process Classification.

GPR Gaussian Process Regression.

GUI Graphical User Interface.

HDP Hochdruckpumpe.

HiL Hardware-in-the-Loop.

HILOMOT Hierarchical Local Model Tree.

HPFS The High Pressure Fuel Supply.

HSFM Heuristic Space Filling Metric.

i.i.d independent and identically distributed.

LGPR Local Gaussian Process Regression.

LLM local linear model.

LOLIMOT Local Linear Model Tree.

MISO multiple input single output.

MLP Multilayer Perceptron.

MSV Mengensteuerventil.

NARX Nonlinear Auto-Regression with eXogenous inputs.

NN Neural Networks.

NOE Nonlinear Output Error.

PDF Probability Density Function.

PRBS pseudo random binary sequence.

PWM Pulse Width Modulated.

SGPR Sparse Gaussian Process Regression.

SISO Single input single output.

VW-LGPR Variance Weighted Local Gaussian Process Regression.

Bibliography

Baumann, W., Röpke, K., Stelzer, S. & Frank, A. (2011), *"Model-Based Calibration Process for Diesel Engines."*, in 4th International Symposium on Development Methodology, pages 196-208, Wiesbaden, Germany.

Baumann, W., Schaum, S., Röpke, K. & Knaak, M. (2008), *"Excitation Signals for Nonlinear Dynamic Modeling of Combustion Engines"*, In: IFAC World Conference 2008, Seoul, Korea.

Berger, B., Rauscher, F. & Lohmann, B. (2011), *"Analysing Gaussian Processes for Stationary Black-Box Combustion Engine Modelling"*, 18th IFAC World Congress Milano (Italy).

Bittermann, A., Kranawetter, E., Krenn, J., Ladein, B., Ebner, T., Altenstrasser, H., Koegeler, H.-M. & Gschweitl, K. (2004), *"Emission Development of Vehicle Diesel Engines by Means of DoE and Computer Simulation"*, MTZ - worldwide,06:15-19.

Boumans, M. (2008), *"Weiterentwicklung eines Hardware-in-the-Loop-Simulationsmodells für den Einsatz im Steuergeräteentwicklungsprozess"*, Master Thesis, Fernuniversität Hagen.

Breiman, L. (1993), *"Hinging Hyperplanes for Regression, Classification, and Function Approximation*, in: IEEE rans. On Information Theory, 39(3).

Bretthorst, G. (1988), *"Bayesian Spectrum Analysis and Parameter Estimation*, Springer-Verlag, New York.

Bretthorst, G. (1990), *"An Introduction to Parameter Estimation Using Bayesian Probability Theory*, in: Maximum Entropy and Bayesian Methods Fundamental Theories of Physics Volume 39, pp 53-79.

Brigham, E. (1974), *"The Fast Fourier Transform"*, Pentice-Hall, Englewood Cliffs, NJ.

Çelik, A. S. (2012), *"Evaluierung von Applikationsprozessen der Motorsteuergeräteentwicklung am virtuellen Versuchsträger am Hardware-in-the-Loop System"*, Diploma Thesis, Universität Stuttgart.

Deflorian, M. & Klöpper, F. (2009), *"Design of dynamic experiments"*, In: Proceedings of the 5th Conference Design of Experiments (DoE) in Engine Development.

Deflorian, M., Klöpper, F. & Rückert, J. (2010), *"Online Dynamic Black Box Modelling and Adaptive Experiment Design in Combustion Engine Calibration"*, 6th IFAC Symposium Advances in Automotive Control.

Deflorian, M. & Zaglauer, S. (2011), *"Design of Experiments for nonlinear dynamic system identification"*, 18th IFAC World Congress.

Ernst, S. (1998), *"Hinging hyperplane trees for approximation and identification*, in: IEEE Conference on Decision and Control (CDC), pages 1261-1277, Tampa, USA.

Fedorov, V. (1972), *"Theory of optimal experiments*, Academic Press Inc. New York.

Fedorov, V. & Hackl, P. (1997), *"Model-oriented design of experiments*, Springer-Verlag.

Gibbs, M. N. (1997), *"Bayesian Gaussian Processes for Regression and Classification"*, PhD thesis, Cambridge University.

Gibbs, M. N. & MacKay, D. J. (1997), *"Efficient Implementation of Gaussian Processes"*, Technical report, Cavendish Laboratory, Cambridge, United Kingdom.

Godward, T., Schilling, H. & Schaum, S. (2013), *"Use of Dynamic Testing and Modeling in the Engine Calibration Process"*, Design of Experiments (DoE) in Engine Development, 7 th Conference.

Gull, S. F. (1988), *"Bayesian inductive inference and maximum entropy"*, In: Maximum Entropy and Bayesian Methods in Science and Engineering, vol. 1: Foundations, ed. by G. Erickson and C. Smith, pp. 53-74, Dordrecht. Kluwer.

Gutjahr, T. (2012), *"Dynamic System Identification with Gaussian Processes in Model-Based Engine Development"*, PhD Thesis, Universität Ilmenau.

Gutjahr, T., Kleinegräber, H., Ulmer, H., Kruse, T. & Eckstein, C. (2013), *"New Approaches for Modeling Dynamic Engine Behavior with Gaussian Processes"*, Design of Experiments (DoE) in Engine Development, 7 th Conference.

Gutjahr, T., Ulmer, H., Kruse, T., Markert, H. & Ament, C. (2011), *"Advanced Approaches for the Identification of Dynamic Engine Behavior with Probabilistic Modeling"*, Design of Experiments (DoE) in Engine Development, 6 th Conference.

Haber, R. (1985), *"Nonlinearity tests for dynamic processes"*, In IFAC Symposium on Identification and System Parameter Estimation, pages 409-414, York, UK.

Hametner, C. & Jakubek, S. (2012), *"Local model network identification for online engine modelling"*, in: Information Science 220, pages 210-225, Elsevier Inc.

Hametner, C. & Nebel, M. (2011), *"Operating regime based dynamic engine modelling"*, in: Control Engine Practice 20, pages 397-407, Elsevier Ltd.

Hametner, C., Stadlbauer, M., Deregnaucourt, M., Jakubek, S. & Winsel, T. (2013), *"Optimal experiment design based on local model networks and multilayer perceptron networks"*, In: Engineering Applications of Artificial Intelligence, Volume 26, Issue 1, January 2013, pages 251-261.

Hartmann, B., Baumann, W. & Nelles, O. (2013), *"Axes-Oblique Partitioning of Local Model Networks for Engine Calibration"*, Design of Experiments (DoE) in Engine Development, 7 th Conference.

Hofmann, S. (2003), *"Identifikation von nichtlinearen mechatronischen Systemen auf der Basis von Volterra Reihen"*, PhD Thesis, TU München.

Isermann, R. (2011), *"Identification of Dynamic Systems"*, Springer Heidelberg Dordrecht London New York.

Jaynes, E. (1986), *"Bayesian methods: General backround"*, In Maximum Entropy and Bayesian Methods in Apllied Statistics,J. H. Justice, ed., pp. 1-25. Cambridge University Press, Cambridge.

Jaynes, E. T. (1985), *"Highly Informative Priors*, in J.M. Bernardo, M.H. DeGroot,

D.V., Lindley, and A.F.M. Smith (eds.), Bayesian Statistics 2, Elsevier Science Publishers, Amsterdam, p. 329.

Knoedler, K. (2004), *"Methoden der restringierten Online-Optimierung zur Basisapplikation moderner Verbrennungsmotoren"*, PhD Thesis, Universität Tübingen, Logos Verlag Berlin.

Kruse, T., Kurz, S. & Lang, T. (2010), *"Modern Statistical Modeling and Evolutionary Optimization Methods for the Broad Use in ECU Calibration"*, In 6th IFAC Symposium Advances in Automotive Control, Munich, Germany.

Kruse, T., Ulmer, H., Kreuzinger, T. & Lang, T. (2012), *"Einsatz von genauen datenbasierten Modellen bei der Applikation von Steuergeräten am HiL-System"*, In: 4th Simulation und Test für die Automobilelektronik, pages 80-91.

Kruse, T., Ulmer, H. & Schulmeister, U. (2007), *"Use of Advanced Modelling and Optimization for Diesel- and Gasoline Engine Calibration"*, In K. Röpke, editor, Design of Experiments (DoE) in Engine Development, volume 4, pages 91-102. Expert.

Kuss, M. (2006), *"Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning"*, Technische Universität Darmstadt.

Ljung, L. (2007), *"System identification - theory for the user"*, Prentice-Hall, Upper Saddle River, NJ.

Loredo, T. J. (1989), *"From Laplace to Supernova SN 1987A: Bayesian Inferenc in Astrophysics"*, In Maximum Entropy and Bayesian Methods, P. Fougere, ed., pp. 81-142. Kluver, Dordrecht.

MacKay, D. (1992*a*), *"A Practical Bayesian Framework for Backpropagation Networks"*, Neural Computation, 4(3):219-269.

MacKay, D. (1992*b*), *"Bayesian interpolation"*, Neural Computation, 4(3):415-447.

Markert, H., Schiepe, C., Streichert, F., Meister, U., Diener, R. & Gutjahr, T. (2011), *"Comparison of Alternative Approaches to Auto-Regressive Modelling of Dynamic Systems"*, Design of Experiments (DoE) in Engine Development, 6 th Conference.

Mitterer, A. (2000), *"Optimierung vielparametriger Systeme in der Kfz-Antriebsentwicklung"*, PhD Thesis, Technische Universität München.

Narendra, K. S. & Parthasarathy, K. (1990), *"Identification and Control of Dynamical Systems Using Neural Networks*, IEEE Transactions on Neural Networks. Vol.1 No.1.

Neal, R. (1995), *"Bayesian Learning for Neural Networks"*, PhD thesis, Dept. of Computer Science, University of Toronto.

Nelles, O. (1997), *"LOLIMOT- Lokale, lineare Modelle zur Identifikation nichtlinearer, dynamischer Systeme"*, Automatisierungstechnik (at), 45(4): pages 163-174.

Nelles, O. (2001), *"Nonlinear System Identification"*, Springer-Verlag Berlin Heidelberg New York.

Nelles, O. (2006), *"Axis-oblique partitioning strategies for local model networks*, in: International Symposium on Intelligent Control (ISIC), Munich, Germany.

Nelles, O., Sinsel, S. & Isermann, R. (1996), *"Local basis function networks for identification of a turbo charger"*, in: IEE UKACC International Conference on Control, pages 7-12, Exeter, UK.

Nguyen-Tuong, D., Bischoff, S., Imhof, V. & Kloppenburg, E. (2014), *"DE 10 2013 206 258 A1, registration date 10.04.2013*, Deutsches Patent- und Markenamt.

Nguyen-Tuong, D., Markert, H. & Meister, U. (2014), *"DE 10 2013 206 297 A1, registration date 10.04.2013*, Deutsches Patent- und Markenamt.

Nguyen-Tuong, D., Peters, J. & Seeger, M. (2008), *"Local Gaussian Process Regression for Real Time Online Model Learning and Control"*, Advances in Neural Information Processing Systems, 22 (NIPS 2008), Cambridge, MA: MIT Press.

Nguyen-Tuong, D., Seeger, M. & Peters, J. (2009), *"Model Learning with Local Gaussian Process Regression"*, Advanced Robotics, pages 2015-2034.

Nguyen-Tuong, D., Seeger, M. & Peters, J. (2010), *"Real-Time Local GP Model Learning"*, From Motor Learning to Interaction Learning in Robots, Springer-Verlag Berlin Heidelberg.

O'Hagan, A. (1978), *"Curve Fitting and Optimal Design for Prediction"*, In Journal of the Royal Statistical Society, volume 40 of B, pages 1-42. JSTOR.

Orfanidis, S. (1996), *"Optimum Signal Processing. An Introduction. 2nd Edition*, Prentice-Hall, Englewood Cliffs, NJ.

Paciorek, C. J. & Schervish, M. J. (2004), *"Nonstationary Covariance Functions for Gaussian Process Regression"*, In: Thrun, S., Saul, L., Schoelkopf, B. (eds.) Advances in Neural Information Processing Systems 16. MIT Press, Cambridge.

Pintelon, R. & Schoukens, J. (2012), *"System Identification: A Frequency Domain Approach"*, 2nd Edition, Wiley-IEEE Press.

Plagemann, C., Kersting, C. & Burgard, W. (2008), *"Nonstationary Gaussian Process Regression Using Point Estimates of Local Smoothness"*, W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 204-219, Springer-Verlag Berlin Heidelberg.

Pucar, P. & Millnert, M. (1995), *"Smooth hinging hyperplanes - an alternative to neural nets*, Proc. 3rd ECC, Italy.

Quiñonero-Candela, J. & Rasmussen, C. E. (2005), *"A Unifying View of Sparse Approximate Gaussian Process Regression*, Journal of Machine Learning Research 6: pages 1939 - 1959.

Rasmussen, C. E. (1996), *"Evaluation of Gaussian Processes and other methods for non-linear regression"*, Ph.D. thesis, Dept. of Computer Science, University of Toronto.

Rasmussen, C. E. & Williams, C. K. I. (2006), *"Gaussian Processes for Machine Learning"*, MIT Press.

Raudies, H. (2012), *"Regressionstests am virtuellen Versuchsträger"*, Master Thesis, Hochschule Mannheim.

Röpke, K., Baumann, W., Köhler, B.-U., Schaum, S., Lange, R. & Knaak, M. (2012), *"Engine Calibration Using Nonlinear Dynamic Modeling"*, in: Alberer, D., Hjalmarsson, H., del Re, L., Identification for Automotive Systems, pages 165-182, Springer-Verlag London Limited.

Santner, T. J., Williams, B. J. & Notz, W. I. (2003), *"The Design and Analysis of Computer Experiments (Springer Series in Statistics)"*, Springer-Verlag New York.

Schlosser, A., Schönefelder, C., Hendrikx, M., Pischinger, S. & Sentis, T. (2009), *"Automated ECU Calibration - Example: Torque Structure of Gasoline Engine"*, In 5th Design of Experiment (DoE) in Engine Development System, number 14, Berlin, Germany.

Schreiber, A., Kowalczyk, M. & Isermann, R. (2011), *"Method for Dynamic Online Identification with Integrated determination of Operating Boundaries"*, Design of Experiments (DoE) in Engine Development, 6 th Conference, pages 120-135.

Schreiter, J., Markert, H., Hanselmann, M., Nguyen-Tuong, D. & Bohne, C. (2013), *"Large Scale Transient Data-based Models for Simulation of Vehicle Power Demand"*, Design of Experiments (DoE) in Engine Development, 7 th Conference.

Sequenz, H. (2013), *"Emission Modelling and Model-Based Optimisation of the Engine Control"*, Fortschr.-Ber. VDI Reihe 8 Nr. 1222 VDI Verlag.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P., Hjalmarsson, H. & Juditsky, A. (1995), *"Nonlinear Black-box Modeling in System Identification: a Unified Overview*, Automatica Vol. 31(12) pages 1691-1724.

Sobol', I. M. (1967), *"On the distribution of points in a cube and the approximate evaluation of integrals"*, USSR Comput. Math. Math. Phys. 7, 86-112.

Tietze, N. (2011), *"Potenzialanalyse und Parametrierung eines Gesamtfahrzeug-Simulationsmodells für die Applikation von Motorsteuergeräten am Hardware-in-the-Loop System"*, Diploma Thesis, Universität Stuttgart.

Tietze, N. & Billand, D. (2012), *"Einsatz von Regressionstests am virtuellen Versuchsträger zur Qualitätssicherung im Applikationsprozess von Motorsteuergeräten"*, 4.AutoTest - Tagungsband, Forschungsinstitut für Kraftfahrwesen und Fahrzeugmotoren Stuttgart.

Tietze, N., Konigorski, U., Fleck, C. & Nguyen-Tuong, D. (2014a), *"Model-based Calibration of Engine Controller Using Automated Transient Design of Experiment"*, In: 14th Stuttgarter International Symposium. pp. 781-799, Springer-Verlag.

Tietze, N., Konigorski, U. & Nguyen-Tuong, D. (2014b), *"Local Gaussian Process Regression for Model-based Calibration of Engine Control Units"*, In: 5th Simulation and Testing for Automotive Electronics, pages 216-236.

von Mises, R. (1964), *"Mathematical Theory of Probability and Statistics"*, Academic Press.

von Rango, J., Schnorbus, T., Kwee, H., Beck, R., Kinoo, B., Arthozoul, S. & Zhang, M. (2012), *"Comparison of Different Approaches for Global Modeling of Combustion Engines"*, In: 4th Simulation und Test für die Automobilelektronik, pages 80-91.

Williams, C. K. I. (1995), *"Regression with Gaussian Processes"*, To appear in Annals of Mathematics and Artificial Intelligence: Models, Algorithms and Applications.

Williams, C. K. I. & Rasmussen, C. E. (1996), *"Gaussian Processes for Regression"*, In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, editors, Advances in Neural Information Processing Systems, volume 8, pages 514-520. MIT Press.

Williams, C. K. I. & Rasmussen, C. E. (2002), *"Gaussian Processes"*, In M. A. Arbib, editor, Handbook of Brain Theory and Neural Networks, pages 466-470. MIT Press, second edition.